SensorAct: A Decentralized and Scriptable Middleware for Smart Energy Buildings

Pandarasamy Arjunan*, Manaswi Saha*, Haksoo Choi†, Manoj Gulati*,
Amarjeet Singh*, Pushpendra Singh*, Mani B. Srivastava†

*Indraprastha Institute of Information Technology (IIIT), Delhi, India
{pandarasamya,manaswis,manojg,amarjeet,psingh}@iiitd.ac.in
†University of California Los Angeles, United States
haksoo@cs.ucla.edu, mbs@ucla.edu

***Abstract*—Buildings, with their different subsystems interacting with diverse occupants, constitute a complex Cyber-Physical-Human infrastructure. Monitoring and controlling this complex ecosystem is essential both for efficient and optimized operations of building subsystems and for influencing the occupant behavior. A critical enabling technology in this case is a middleware system for buildings that can provide support for deriving rich inferences by fusing and analyzing intentionally acquired or opportunistically available data from diverse embedded sensors, human feedback, and existing building subsystems. This paper presents *SensorAct*, a decentralized and scriptable middleware system architecture for developing and scheduling various energy management applications for smart buildings. In addition to providing support for managing and integrating heterogeneous sensing and actuation systems in buildings, *SensorAct* provides two emerging features: 1) a *scripting framework* for extending and automating the energy management functions of the modern buildings, and 2) a *rule-based sensor data and control sharing* mechanism for fine-grained sharing for building owners. We describe the detailed system architecture and design, and provide proof of concept through multiple third party applications built using *SensorAct* APIs and deployment in diverse settings across India and United States. *SensorAct* is released in open source for community use.**

***Keywords*-Building Management System, Middleware, Energy monitoring, Internet of Things**

I. Introduction

Buildings account for a significant proportion of overall energy use in both the developing (e.g., 47% of total energy in India [12]) and the developed (e.g., 41% in US [2]) countries. Energy consumption in buildings is spread across diverse devices, appliances and control subsystems including lighting, Heating Ventilation and Air Conditioning (HVAC), security and access control. Interdependence of these subsystems, together with occupancy of buildings by diverse users, make buildings a complex *Cyber-Physical-Human* system. Detailed understanding of different subsystems within buildings, together with their interactions with diverse occupants, is critical for developing solutions that optimize overall operations of buildings and scale across diverse usage patterns.

Commercial buildings often employ Building Management Systems (BMS) for monitoring and controlling their subsystems. While striving to achieve an optimal energy efficient control, these BMSes restrict the management to a central facility department. To support interoperability, many of these BMSes support Building Automation Controller Network (BACnet)[1]. However, external third party application development using BACnet interface is complex; hence it restricts widespread development of such applications [15]. Such third party applications, if facilitated, can further support integration of different building sub-systems for improved operations. For example, information from RFID based access control system can be used to infer the occupancy and accordingly control the HVAC and lighting systems.

At the other end of the spectrum, for residential buildings, several home monitoring and automation systems have emerged recently. These systems typically involve monitoring ambient parameters (e.g., motion and temperature) of the home, together with smart metering solutions for detailed logging of whole-home electricity, gas, and water consumption. Observations from such monitoring systems are often used to control various home appliances, e.g., thermostats, water heating, and lighting. Correspondingly, several cloud-based Internet of Things (IoT) and Machine to Machine (M2M) communication platforms have been proposed recently[2]. These platforms allow users to connect their devices, including both sensors for monitoring and actuators for controlling, to a centralized service, and build their own applications using the observed parameters and available actuators.

While these existing systems are a step forward towards developing new applications for monitoring and controlling the energy usage in buildings, their centralized architecture results in several limitations as follows:

1) Data privacy: Detailed monitoring in both commercial and residential buildings results in large volumes of data that can be interpreted to infer several personal attributes [16], e.g., home occupancy or wake-up times. Privacy concerns emerging from such data results

[1] http://www.bacnet.org

[2] http://xively.com, http://www.nimbits.com, http://open.sen.se

978-1-4673-7211-4/15 $31.00 © 2015 IEEE
DOI 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.27

in users refraining from publishing their data to a centralized cloud service [18].

2) Intermittent network connectivity: Poor Internet connectivity, especially in the context of developing countries [5], are de-motivating the dependence on always connected devices to cloud-based IoT platforms[2], specifically for the buildings domain.
3) Real-time control: Data collection and control actions occurring in a cloud are further associated with latency issues arising out of network delays and high usage scenarios. Some operations within buildings, e.g., turning on the sprinkler in case of fire, need real time control that may be difficult to obtain through cloud IoT services.

Motivated by aforementioned limitations, in this paper we present *SensorAct* - a decentralized and scriptable middleware architecture specifically designed for detailed monitoring and control in buildings. Beyond connecting devices and thereby monitoring the built spaces, *SensorAct* provides emerging capabilities including:

1) *Virtual Personal Device Servers (VPDS)* (See Section II-B) for local-hosting of middleware within the buildings to alleviate data privacy, control security and intermittent network connectivity problems.
2) Decentralized and distributed management of building resources involving diverse devices and different stakeholders, including the occupants, at scale. Such decentralized architecture (keeping the middleware closer to the devices) facilitates real time control required for energy management applications.
3) Fine-grained selective sharing of sensor data and control with users at the global scale to alleviate control security concerns and providing building-wise local storage for protecting the data privacy.
4) *Scripting framework* (See Section III), simple programming abstractions for extending the system features and developing energy managing applications involving sensing and control, analysis, alerts, and notifications in rich form to support diverse usage scenarios.

Together with supporting diverse usage scenarios, *SensorAct* accommodates a rich ecosystem of existing and new monitoring and controlling systems. Correspondingly, the key contributions of this paper are:

- A working middleware system, released in open source, that can run on heterogeneous platforms, allowing participants to collect data from a variety of sensors and perform actions thereof.
- A *scripting framework* to extend the middleware system functionality with customized application logic.
- A rule-based *fine-grained access control* mechanism enables sharing of sensor data and actuation control with other participants.

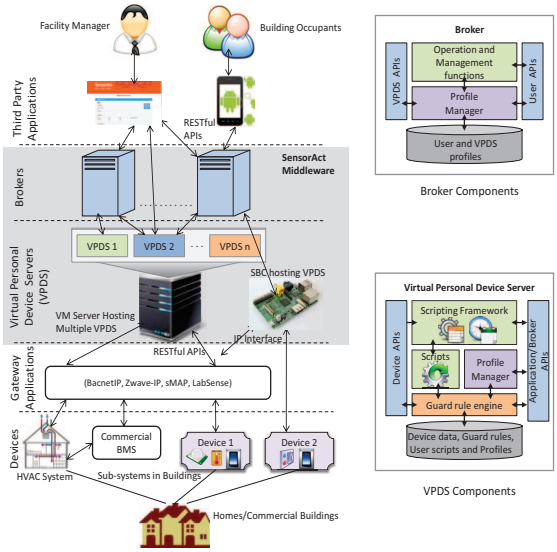

Figure 1: Tiered architecture of *SensorAct* and its various subsystem components.

- Detailed evaluation of the system through extensive deployments involving different usage scenarios and energy monitoring applications.

A preliminary design of *SensorAct* was presented earlier in [4]. Significant system development leading to multiple real world deployments and extensive evaluation of the scripting framework for various energy monitoring applications, are all new inclusions of this paper.

II. SensorAct Architecture

SensorAct adopts a *tiered architecture* connecting the building subsystems with its occupants as shown in Figure 1. The main layers of *SensorAct* are *Virtual Personal Device Servers (VPDS)* and *Brokers*, which interact with *Devices* that monitor and control different building spaces at the bottom layer, and third-party *applications and users* at the top layer.

A. Devices and Gateways

Existing building subsystems contain numerous sensing and actuation points connected using diverse protocols such as BACnet and Modbus. *SensorAct* maps the existing and new sensing and actuation points as *Devices* by eliminating the underlying complex naming conventions used in the legacy building subsystems. A device in *SensorAct* consists of a collection of sensors (monitoring the ambient environment or the state of a building subsystem) and actuators (that allow for changing the state of a building subsystem). Each sensor, in turn, may consist of a collection of channels which measure a particular building phenomenon. *SensorAct* uses an intuitive and hierarchical naming scheme for managing and identifying the devices and their associated sensors/actuators, channels and readings uniquely. As an example, a single reading from a smart energy meter connected with the main panel is identified as

$OwnerName/MainPanel/EnergyMeter1/power/ < timestamp > / < value >$.

Devices are assumed to either communicate directly or through a gateway (e.g., LabSense, as discussed in Section IV-A) that can bridge the sensing interface with *SensorAct* using its RESTful API. Each device is associated with a device profile that contains all meta information. A building owner can manage multiple devices owned by him through the *profile manager* facilitating the creation of device profiles. It consists of a set of key-valued paired attributes such as its name, IP address, a collection of sensor and/or actuator profiles, number of channels, data types and units, location and placement.

B. Virtual Personal Device Server (VPDS)

VPDS is the primary core component of *SensorAct* middleware architecture. As a package, VPDS contains (a) *Data Archiver* for storing and retrieving time-series sensor data, (b) *Scripting Framework* for executing the custom building applications, (c) *Guard Rule Engine* for access control, and (d) *Profile Manager* and APIs for devices, applications, and brokers to interact with the VPDS, as shown in Figure 1. VPDS Owner (VO) may allow other users to have controlled access to the devices registered with the VPDS. These devices monitor and control the buildings owned by VO. Each VPDS has an associated auto-generated owner's key which is used while registering the VPDS with a Broker for data and control sharing.

The VPDS abstraction not only permits flexible provisioning of hardware resources but also allows diverse deployment scenarios including a VPDS hosted on a local machine for high frequency sensors, low dependence on Internet connectivity, high-security and low-latency sensing and control. Such multiple VPDS instances are coordinated through Brokers in the higher tier. VPDS-Broker architecture further enables global access through distributed registry of users at scale.

C. Broker

In *SensorAct*, a trusted broker contains a registry of users, a registry of VPDSes, and it acts as a mediator to assist client applications establish a connection with multiple VPDSes. A VO needs to register her VPDS on one of the brokers to share data and control with other users registered on the corresponding broker. For data communication to scale across multiple VPDSes and users managed by a single broker, direct communication over a secured channel is provided between users and VPDSes. More details about sensor data and control sharing is explained in Section II-E.

D. Applications

Programming abstractions in *SensorAct*, using RESTful APIs, allow easy development of third party applications that provide a user with controlled access to sensor data and actuators. A *SensorAct* user may have two roles: i) An owner of her own devices and their correspondingly

```
{
  NAME: "ReduceResolution",
  PRIORITY: 1,
  TARGET_OPERATION: "READ",
  TARGET_CONSUMERS: [ "alice@sensoract.edu" ],
  CONDITION: "NOT $(WORK_TIME)",
  ACTION: "DownSample(avg, 15min)"
}
```

(a) Guard rule with macros as condition

```
{
  NAME: "WORK_TIME",
  VALUE: "[ * * 9-18 * * 1-5 ]"
}
```

(b) Macro definition

Figure 2: An example of a guard rule and a macro for selective sharing of sensor data.

associated VPDS ii) A user with data and control access as per the privileges assigned by the owner of other VPDS. A single user could be the owner of her own VPDS and user for some other VPDS. As an owner, a user may grant controlled access to other users thus letting them access sensor data from her devices or to even control them in a constrained manner. Third party applications may provide users with a more convenient interface to the underlying functionality of VPDS and broker. We discuss three such third party applications in Section IV.

E. Guard Rule Engine for Selective Sharing

Guard Rule Engine in *SensorAct* is designed to support *selective sharing* of sensor data and actuation control with other users as governed by the corresponding owner. All access requests for actuators or sensor data are governed by Guard Rule Engine. Tight control on the access is maintained through owner-defined *guard rules* which are policies for restricting access to the data and control of the devices configured for buildings. Guard Rule Engine enables *fine grained access control* by allowing the owner to define rules based on user, group, time, location, sensor data, and actuators. Every device registered with the VPDS has an associated set of guard rules, created by corresponding VO, for facilitating external (or shared) access.

The specification of the guard rules and its features such as macros, templates, and built-in functions can be found in [4]. Figure 2 shows an example of a guard rule and a macro. This guard rule enforces a policy to share data during non-working hours only, which is defined by the macro using a *Cron* time expression, and reduces the data access resolution to 15 minutes. The guard rule itself does not necessarily contain references to specific users, sensors, or actuators, but it can be associated to them later by its owner. This *lazy-association* allows users to reuse same rules for different users, multiple devices, or groups of devices. More details on sharing process through Guard Rule Engine can be also found in [4].

III. Scripting Framework

The *scripting framework* in *SensorAct* provides an application execution environment within the middleware for high-level scripting languages in a sandbox. Unlike other systems, wherein external applications read sensor data and perform control actions outside the middleware, this framework enables building owners to inject their custom application logic written in a high-level scripting language,

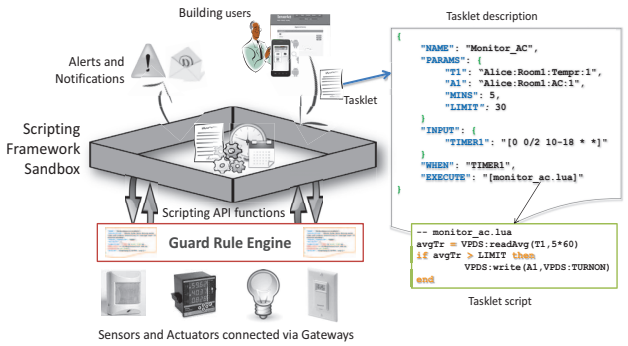

Figure 3: Scripting framework workflow with an example Tasklet.

termed as *tasklets*, into the middleware to perform sophisticated energy management and control operations. These tasklets can be scheduled to read and process live sensor data streams. The proposed *Scripting Framework* also provides a set of read functions to access sensor data streams and write functions to control the actuators. These read/write primitives can then be used to develop complex (one-shot or persistent) control actions providing *rich support for automation* including sensor data processing, data fusion, actuator control, and notifications. Figure 3 illustrates the workflow of the proposed scripting framework.

A. Tasklet Workflow

A *tasklet* in the scripting framework is a piece of light-weight non-blocking script that performs a particular set of operations. As shown in Figure 3, it consists of *tasklet description* and *tasklet script*. While the *tasklet description* contains meta information about the tasklet, such as resources (sensors and actuators identifier) to be used, parameters, scheduling and triggering conditions, *tasklet script* contains the application logic in a high-level scripting language. More details about tasklet description can be found in [4]. A *tasklet scheduler* receives the tasklet execution requests, submitted using the corresponding tasklet management API provided by the VPDS, and it is responsible for scheduling, managing, and controlling the tasklet throughout its life-cycle. Once a tasklet is scheduled for execution, the *tasklet scheduler* first classifies and schedules the tasklet, based upon the identifiers used in the *When* primitive of the description, as one of the following categories:

One-shot: One-time and immediate execution of a tasklet script. For example, querying the current status of a sensor or instantaneous switching of an actuator.

Periodic: Executing a tasklet script whenever timer elapses to perform periodic operations. For example, switch on my office air-conditioner at 9AM only on weekdays or email me the electricity usage summary every day at 8PM.

Event based: Executing a tasklet script whenever a change in the value of sensors or actuators is observed. For example, switching off lights when a window is opened.

```
map    read(DeviceId, duration(in seconds))
map    read(DeviceId, startTime, endTime)
number read(DeviceId, duration, [sum|count|min|max|mean] )
number read(DeviceId, startTime, endTime, [sum|count|min|max|mean] )
string plot(DeviceId, duration)
boolean write(DeviceId, status=ON|OFF|value)
boolean email(to, subject, message [,plot])
boolean sms(to, message)
```

Table I: Primitive API functions in Scripting framework available to use in Tasklet Scripts.

Tasklet manager provides an isolated execution environment for each tasklet and restricts any interaction among them for sensitive building control applications. By default, a tasklet inherits the privileges of a user who submitted it. All read and write operations on sensors and actuators performed from tasklets go through Guard Rule Engine. Correspondingly, tasklets can only perform operations allowed for a user invoking them, thus protecting against unauthorized access.

B. Tasklet API Functions

The scripting framework in *SensorAct* provides a set of primitive API functions as shown in Table I, in order to perform various runtime operations. Tasklet scripts can invoke these low-level API functions and it enables developers to create and schedule custom building control and automation applications. While expert users can write complex tasklet scripts on their own, novice users can use an interactive web interface to create simple automation applications. We also plan to provide *tasklet templates* for commonly used applications, so that users can easily create tasklets by filling in only the required parameters.

The proposed tasklet framework provides a platform to implement and schedule several energy management tasks within the middleware system. Examples are 1) inferring high-level rich occupant-specific context information, such as occupancy and usage patterns, by fusing several raw sensor and actuator values; 2) automation of the routine building control activities performed by the occupants in their day-to-day life, e.g., pre-heating or pre-cooling the workspaces in advance; 3) custom creation of *coordinated appliances* based upon detected building events, e.g., switching off a meeting room may result in switching a group of devices together or in a particular sequence; and 4) development of an automated alert or notification system for monitoring and management of building premises, e.g., alerting the facility management team in case of any abnormal energy consumption. In Section V, we present a list of energy management applications created using the proposed tasklet framework.

IV. Implementation

A. Gateways and Devices

Gateways interconnect existing and new sensors and actuators in the buildings with a VPDS. In the current implementation, three gateways are supported: 1) LabSense[11] for interfacing Z-Wave based ambient sensors (temperature, light intensity, motion, and door contact status) in existing Home Automation Systems, Modbus based smart energy

Component	VPDS API endpoints
User	/user/{register|list}
Device	/device/{add|delete|get|list}
	/device/template/{add|delete|get|list}
Guardrule	/guardrule/{add|delete|get|list}
	/guardrule/association{add|delete}
Tasklet	/tasklet/{add|delete|get|list}
	/tasklet/{execute|cancel|status}
Data	/data/{upload/wavesegment|query}
Share	/device/share
Component	**Broker API endpoints**
User	/user/{register|login|list}
VPDS	/vpds/{register|get|list}
Device	/device/{search|share}
	/device/{user|owner}/shared

Table II: A list of APIs supported by different components of the *SensorAct* architecture.

meters, SNMP based Raritan[3] power distribution unit, and Modbus based Veris and Eaton[4] energy meters, 2) sMAP [9] to communicate with several commercial energy meters and HVAC systems using different protocols such as Modbus and BACNet, and 3) A custom built Wi-Fi based Flyport[5] module to interface ambient sensors and actuation relays. These gateways push sensor readings in a uniform format, as described in [4], to VPDS using RESTful APIs, shown in Table II, and execute the actuation commands received from the building control applications.

B. VPDS and Broker

Various VPDS and Broker components, as explained in Figure 1, were implemented in Java using open source tools and the code was released for community use[6]. *SensorAct* exposes a rich set of RESTful APIs for most of its functionalities. Such open APIs enable easy integration with other systems and allow developers to write custom third party (stand-alone, web, or mobile based) applications for extending the system features, e.g., visualization, scripting, access control, and sharing. Table II lists the APIs currently implemented for various components in VPDS and Broker.

The scripting framework in *SensorAct* uses Quartz[7] library to schedule and execute *tasklets*. The current implementation supports Lua[8] and Jython[9] to write tasklet scripts. Lua was chosen because it is a light-weight, compact, and easy-to-learn scripting language which also has been widely used to program home automation systems. Jython, an implementation of the Python scripting language in Java, provides rich support for data processing. Further, a VPDS instance can be hosted on multifarious devices including single board computers such Raspberry Pi.

C. Applications

Three third-party applications were implemented using the VPDS and Broker APIs and they are explained below.

[3]http://www.raritan.com/
[4]http://www.veris.com, http://www.eaton.com
[5]http://www.openpicus.com
[6]http://github.com/iiitd-ucla-pc3
[7]http://quartz-scheduler.org
[8]http://www.lua.org
[9]http://www.jython.org/

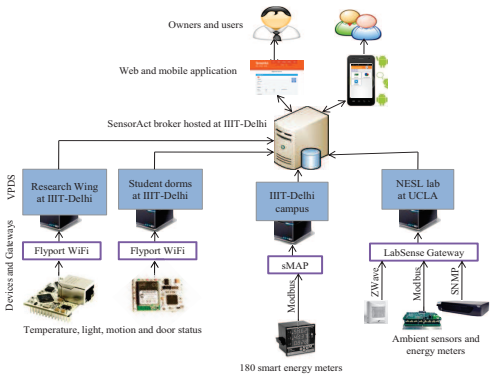

Figure 4: Various deployment scenarios of *SensorAct* architecture. Four VPDSes deployed across different locations were connected to a common broker hosted in IIIT-Delhi.

Web portal: It is a configurable stand-alone web application that interacts with a broker and registered VPDSes using the *SensorAct* APIs. Both VPDS owner and other users can use this application to manage devices, guard rules, and tasklets based upon granted privileges. An integrated plotting application also enables users to visualize their sensor data.

Time and presence based actuation: It is an intuitive web interface, extended from the web portal application, which allows users to actuate their devices remotely based on time and presence, e.g., pre-heating/cooling a workspace. The user interface makes use of tasklets to allow users to switch their appliance "now", "once" at a specific time, or "periodically" at a regular time interval. Guard rules are used to restrict users to actuate the devices in their own spaces only. This system is currently under deployment in a commercial building for lighting control.

Mobile application: An Android application was developed whereby users can specify their Broker credentials and correspondingly manage the devices owned or shared with them.

Further, in order to simplify the *SensorAct* system installation for a building, all VPDS, broker and user interface components of *SensorAct* were packaged into a Virtual Machine image. Detailed installation instruction manual were documented and the usage of the system was evaluated using an user study (See Section V-C).

V. Evaluation

The proposed *SensorAct* architecture is evaluated based on multiple real-world deployments for supporting different usage scenarios. Particularly, the utility of the proposed tasklet framework is shown for various energy monitoring and alerting applications. Further, a user study was performed to evaluate the different aspects of deployment of *SensorAct* in a student dormitory building.

SensorAct was deployed in four different settings as illustrated in Figure 4: (1) Campus wide energy monitoring at IIIT-Delhi, India, (2) Student dormitory deployment at IIIT-Delhi, India, (3) Research lab in UCLA, USA, and (4)

Deployment	Research Wing	Student Dorms	IIITD Campus	NESL, UCLA
Purpose	Occupancy sensing and data sharing	Occupancy sensing and data sharing	Energy monitoring and alerts	Occupancy sensing and energy monitoring
Scale	Single building	21 student dorms	6 buildings	Research lab
Platform	Virtual machine	Laptop and PCs	Virtual machine	Virtual machine and MiniITX
Sensors	Ambient sensors (14)	Ambient sensors (21)	Electricity meters(180)	Ambient sensors and electricity meters(3)
Sampling rate	1 second	1 second	30 seconds	2 seconds
Gateway & protocols	FlyPort, WiFi	FlyPort, WiFi	sMAP, Modbus	LabSense, ZWave, Modbus
Duration	2 months	1 month	4 months	8 months
Users	20	21	2	15

Table III: Different deployment details of *SensorAct* system. Ambient sensors include Temperature, light intensity, motion and door contact status

Research wing at IIIT-Delhi. Each deployment was done with different requirements, devices, gateways, users, and scales as shown in Table III. Separate VPDS instances were used for each deployment, and they were registered with a common broker, hosted at IIIT Delhi, for sharing sensor data and control with users across different buildings. In each deployment, the corresponding owner or tenant of the building managed the VPDS and granted privileges to other occupants who all were registered themselves with the common broker, if required.

A. Campus Wide Electricity Monitoring at IIIT-Delhi

One of the largest deployments of *SensorAct* involved monitoring the electricity usage of all the buildings in IIIT-Delhi campus. IIIT-Delhi campus was newly constructed two years ago in a space of 25 acres. It consists of five buildings: academic, facilities, faculty apartments (30 flats), mess and hostel (400 dorm rooms) buildings. All these buildings are equipped with a commercial BMS system for managing the various building operations, under the administration of a facility manager (FM). In addition to the commercial BMS system, all the buildings (each floor and flats) were instrumented with over 180 smart meters measuring various electrical parameters. A sMAP based archiver was used for collecting meter readings at every 30 seconds. Existing BMS subsystems, such as HVAC, were also interfaced with the sMAP archiver using BACnet and Modbus bridge. A separate sMAP to *SensorAct* gateway was implemented for uploading all real-time measurements to *SensorAct*. As per FM's requirements, three energy management applications were created, to be managed by him, using the proposed tasklet framework.

1) Abnormal street light usage detection: IIIT-Delhi campus contains pathways around the campus for about 2 kilometers. There are about 135 street lights installed in the path way. These street lights are manually switched on in the evening and switched off in the morning by an operator. They consume over 6 kilo-watts of power. From the street light meter readings, as shown in Figure 5, we observed that there were some suspicious electricity usage events during day time and occasional events during night time as well.

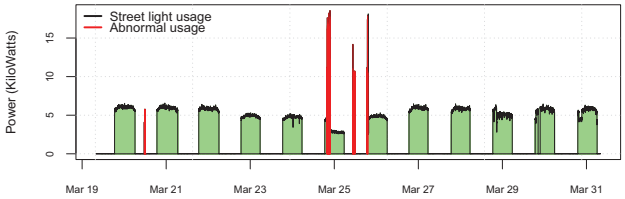

Figure 5: Daily electricity usage pattern for street lights (6:30pm to 6:00am every day) for 12 days and some abnormal energy usage events (marked in red).

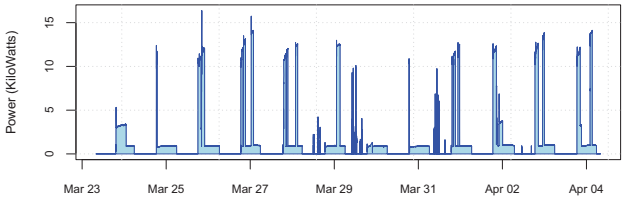

Figure 6: Electricity usage patterns of sports area lights for 12 days. Few street lights around the sports area consuming about 0.9 kilo-watts of power are used from 6pm to 6am every day. Each spike in this plot corresponds to the usage of flood lights in the sports area.

Hence, to monitor such abnormal electricity usage events by street lights, two tasklets, one for day time and another for night time were setup. They computed average electricity consumption of the street lights every five minutes. If the average consumption was above a threshold (derived based on our observation), these tasklets sent an email and SMS to the FM for taking necessary action. Over the course of past one month of this setup, these tasklets detected two such abnormal electricity usage events and notified the FM. The FM asked the facility support team to check the street lights and its power meters. FM also suggested that such period monitoring tasklets for street lights are essential, particularly in India, as electricity theft is not uncommon.

2) Sports area usage summary: IIIT-Delhi campus has a sports area that consists of a basket ball court, a foot ball court and a common play ground. The entire sports area is equipped with several flood lights and they consume over 10 kilo-watts of power when they are in use. At present, students are advised to turn on and off these lights whenever

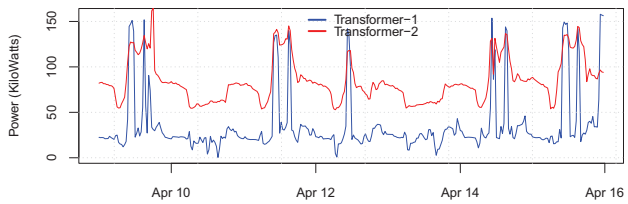

Figure 7: Campus-wide total commercial electricity usage (from two transformers) for 1 week. The spikes in transformer 1 corresponding to the electriciity consumption of HVAC systems, that consusumes over 100 kilo-watts.

they want to play during night time. For accountability and to make policies for the sports area usage, FM wanted to know how many hours these sports area was being used every day and the corresponding energy usage.

Since the sports area lights were connected through a separate smart meter, we monitored the electricity usage for a month to know the baseline usage. Figure 6 illustrates power consumption of the sports are lights (all the peaks) along with some other constant load for 12 days. Based on our observation, a periodic tasklet was created and it was scheduled to run at 8 A.M. everyday. The tasklet reads the smart meters readings for the previous night and filtered out the readings only for the sports area usage based on a known threshold value. The tasklet is also configured to send a summary email to the FM about how many hours the sports area was used and the total energy consumption.

3) Critical energy usage alert: IIIT-Delhi campus receives two power lines from the grid, one for commercial and another for residential usage. While the residential power line is being used for faculty apartments and student's hostels, commercial power line is being used by the rest of loads in the campus. External commercial load from the grid is stepped down using two transformers, one for supplying high-voltage loads such as HVAC, and another one for connecting commercial usage such as lighting and IT devices. As shown in the Figure 7, commercial energy usage for a typical working day is over 3,000 units and it varies based on many other factors. In order to monitor the overall energy usage, FM wanted to develop an application that can alert him when the total energy consumption of the current day exceeds the previous day.

A periodic tasklet was created for monitoring the campus wide energy usage in real time. The tasklet was scheduled to be run at every five minutes. The tasklet script was configured to perform the following tasks in each run: 1) it reads the previous day and current day energy usage from the smart meters which are connected with the corresponding transformers, and 2) it compares them and sends an email and SMS to the FM if the current day energy usage exceeds the previous day consumption, 3) Since this is a periodic tasklet and to avoid continuously sending the notification on successive alerts, it was configured to send alerts two times maximum in a day.

B. Research Lab at UCLA

In this deployment, *SensorAct* was deployed for a 1,200 square feet university research lab, occupied by 12 graduate students, in UCLA for occupancy and electricity monitoring applications. It was instrumented with several sensors including 1) high frequency multi channel Raritan, Eaton, and Veris power meters measuring several electrical parameters of the lab servers, devices, and power outlets, and 2) Z-Wave based Aeon Labs door sensors and HomeSeer multi-sensor (measuring motion, temperature, and light intensity level) was interfaced with Mi Casa Verde Vera[10]

LabSense[11] gateway was used to pull data from all of these sensors and push it to *SensorAct*. FireSense[12] system was also used to monitor the real time network traffic of the lab and the network event logs were pushed to *SensorAct* in real time. Permission to access sensor data and create tasklets was shared with one of the graduate student at IIIT-Delhi, who created a computed sensor for presence detection (discussed in Section IV) and performed occupancy based experiments.

Occupancy and energy usage summary: Based on this setup, a periodic tasklet was created for monitoring the occupancy and electricity usage of the lab. The tasklet was scheduled to run at 12 A.M. midnight every day to send a summary report to the lab members. The tasklet script was primarily doing three functions: 1) it decided whether some one is entering or exiting the lab by fusing the door status (open/close) with motion sensor events, 2) it compares the entry and exit events of the two doors and decides the first entry and last exit time of the lab, 3) it aggregated the total power consumption of a day and created a plot of power consumption at 5 minutes, and 4) finally, it sent a summary email mentioning how many hours lab was opened on a particular day and how was the total power consumption during that time. This energy usage summary report was useful for giving insights about the usage patterns of the lab and for increasing energy usage awareness of the occupants. Further, the inferred occupancy information can also be used to pre-heat/cool the lab.

C. Student Dormitory Deployment at IIIT-Delhi

In this deployment, in order to validate the utility and easy deployment of *SensorAct* middleware system, 21 student groups (each with 2 students at IIIT-Delhi were engaged. They deployed Flyport based Wi-Fi nodes in student dormitory rooms, collecting motion, temperature, and window status information every second, by following the installation manual. One of the faculty coordinator hosted a central VPDS on one of the servers and registered it with the broker at IIIT-Delhi. Students were then asked to register themselves and room occupants (for the case when they are

[10]http://micasaverde.com, http://www.homeseer.com
[11]https://github.com/nesl/LabSense
[12]https://github.com/nesl/FireSense

deploying in someone else's room) on the broker. Faculty coordinator first shared privileges with the engaged student groups and allowed them to create devices in his VPDS though they were not allowed to see data from the devices added by them. This privilege was revoked after two days, and the added devices were shared with the corresponding room occupants to let them decide whom they want to share data. While, the study mandated each student to collect only 2 days worth of data from their individual deployment, a total of 100 days worth of data together from all the groups was reported by the students.

A survey was conducted at the end of the deployment to get feedback from these students on different aspects of using the *SensorAct* system. Overall 17 student groups responded to the survey at the end of the study. Among them 16 student groups had no prior experience with uploading data to a server or cloud system such as *SensorAct*. More than 80% of the respondents mentioned that *SensorAct* installation and configuration on their individual laptops, with different OS, was easy. Approximately 45% of respondents gave their preference for local hosting of *SensorAct* VPDS instead of cloud, due to privacy reasons, if they were to deploy *SensorAct* for monitoring and control in their homes. Students used *SensorAct* for basic data collection and visualization while the occupants used sharing capability for data and control to provide access of their devices to their friends. 64% of them found *SensorAct* to be a good and usable system. More than 90% of responses rated *SensorAct* documentation to be detailed enough with 73% of them asserting that, with the current level of documentation, a new person can setup *SensorAct* system without any help. More than 90% of the responses were positive about the usability of the browser application that was used as a front end for the study.

VI. Related Work

A. Building Management Systems

Several commercial building management systems (BMS)[13] and home automation systems[10] are currently available in the market for monitoring, controlling, and automating various building subsystems and operations. Typically, they comprise of several isolated subsystems each performing a particular task such as fire alarms, security and access control and HVAC. While they include a large number of sensing points spread across a building, data from these sensors are usually inaccessible to building occupants as these systems are normally controlled and managed by a central facility department. While many commercial buildings already have some form of BMS in existence, *SensorAct* can be used to augment them to develop novel occupant-centric applications such as personalized control of workspaces. Gateway applications can be easily developed to interface such M2M applications with *SensorAct* (as shown in Figure 1). Further, higher costs typically associated with such BMSs, prohibit their usage across small deployments such as in residential homes. Home automation systems, try to fill in the gap providing comfort to the occupants, using local storage and several automation scripts. However, they provide limited support for fine-grained data and control sharing across multiple homes and users, supported extensively in *SensorAct*.

[13] http://www.trane.com, http://www.johnsoncontrols.com

B. Research Systems and Architectures

Several research systems pertaining to connecting and sharing sensors at Internet scale, such as SenseWeb [13], SensorWeb [8], GSN [1], and WattDepot [6] have been developed and deployed in the recent past. However, these systems are limited primarily to sensory data aggregation and visualization and provide minimal sharing capabilities. Some research software systems have been proposed in the literature that provides an abstraction over diverse devices and enables uniform interfaces to access them [14], [11]. For example, HomeOS [11] addresses the interoperability and usability issues by providing a PC like abstraction over the networked devices as peripherals for both users and developers. Sensor Andrew [17] shares some common design goals with *SensorAct*. It focuses on a large scale data aggregation from diverse sensors and event-based control for building operations. However, the data and control sharing mechanism is at coarse-grained level, based only on user identity. BuildingDepot [3] focuses on managing network of buildings by isolating the data, users, and privilege management from each other. Similarly, Building Operating System Services (BOSS) and Building Application Stack [10] enable writing portable and fault-tolerant applications on top of diverse physical resources present in buildings. Specific to residential buildings, VHome [18] provides an isolated application execution environment for data analysis. Though VHome shares several design goals with *SensorAct*, it provides an application runtime to execute Cloud Based Application, focusing only on energy data analytics. Further, access control mechanism in VHome supports only time and location based energy data sharing, whereas *SensorAct* supports fine-grained sharing.

C. Cloud-based IoT platforms

Several public cloud-centric IoT platforms exist today for collecting, archiving and visualizing the real time sensory data such as Xively, Nimbits, and Sen.se. While these services provide rich support for data aggregation and visualization of sensory data collected from diverse devices, they provide inadequate or limited support for IoT applications specifically pertaining to the buildings domain: 1) They provide limited capabilities for sharing the collected sensory data as they follow "all-or-nothing" model based only on user identity. Since the sensory data collected from buildings carry several forms of occupancy and usage patterns about the buildings, devices and occupants [7], controlled sharing is required to protect the sensitive sensory data collected

from buildings. 2) They provide limited or no support for controlling the devices and automating their operations. A few services such as Sen.Se provide remote actuation support but they handle simple use cases wherein control is manual or is based on events, e.g., whenever motion is detected, switch on/off an appliance. *SensorAct* architecture allows for easy support of complex energy management applications using the scripting framework. In addition to the features provided by these systems, *SensorAct* supports rule-based selective sharing model for sensory data.

VII. Conclusions

In this paper, we presented the design and development of *SensorAct* - an open source distributed and scriptable middleware system for energy management in buildings. *SensorAct* architecture supports several novel features including (i) Virtual Personal Device Servers (VPDS) for local hosting the middleware within the building, (ii) Scripting framework within the middleware for providing rich support for developing and automating energy management applications, and (iii) A rule-based *fine-grained access control* mechanism enables sharing of sensor data and actuation control with other users. Validation of the developed system was done using multiple deployments, from residential to commercial buildings, spread across India and USA.

Acknowledgment

This work is partially supported through Indo-US PC3 collaborative program, supported by NSF, USA (Grant Number CNS-1143667) and DEITy, India (Grant Number DeitY/R&D/ITEA/4(2)/2012). Authors will also like to acknowledge the support provided by ITRA project, funded by DEITy, India (Reference Number ITRA/15(57)/Mobile/HumanSense/01). We also acknowledge IBM Research, India for partial support of first author through IBM PhD Fellowship. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

References

[1] K. Aberer, M. Hauswirth, and A. Salehi. Global sensor networks. *EPFL, Lausanne, Tech. Rep*, 2006.

[2] AEO. US Energy Information Administration. *AEO2011: Annual Energy Outlook*, April 2011.

[3] Y. Agarwal, R. Gupta, D. Komaki, and T. Weng. Buildingdepot: an extensible and distributed architecture for building data storage, access and sharing. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 64–71. ACM, 2012.

[4] P. Arjunan, N. Batra, H. Choi, A. Singh, P. Singh, and M. B. Srivastava. Sensoract: a privacy and security aware federated middleware for building management. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 80–87. ACM, 2012.

[5] N. Batra, M. Gulati, A. Singh, and M. B. Srivastava. Its different: Insights into home energy consumption in india. In *Proceedings of the Fifth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, BuildSys '13, 2013.

[6] R. S. Brewer and P. M. Johnson. Wattdepot: An open source software ecosystem for enterprise-scale energy data collection, storage, analysis, and visualization. In *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, pages 91–95. IEEE, 2010.

[7] H. Choi, S. Chakraborty, Z. M. Charbiwala, and M. B. Srivastava. Sensorsafe: a framework for privacy-preserving management of personal sensory information. In *Secure Data Management*, pages 85–100. Springer, 2011.

[8] X. Chu, B. Durnota, R. Buyya, et al. Open sensor web architecture: Core services. In *Intelligent Sensing and Information Processing, 2006. ICISIP 2006. Fourth International Conference on*, pages 98–103. IEEE, 2006.

[9] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler. smap: a simple measurement and actuation profile for physical information. In *Proc. of SenSys*, pages 197–210. ACM, 2010.

[10] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. Culler. Boss: Building operating system services. In *Proc. of NSDI*, 2013.

[11] C. Dixon, R. Mahajan, S. Agarwal, A. Brush, B. Lee, S. Saroiu, and V. Bahl. An operating system for the home. In *Proc. of NSDI*. ACM, 2012.

[12] M. Evans, B. Shui, and S. Somasundaram. Country report on building energy codes in india. In *Pacific Northwest National Laboratory, PNNL-17925*, April 2009.

[13] W. I. Grosky, A. Kansal, S. Nath, J. Liu, and F. Zhao. Senseweb: An infrastructure for shared sensing. *Multimedia, IEEE*, 14(4):8–13, 2007.

[14] X. Jiang. A High-Fidelity Energy Monitoring and Feedback Architecture for Reducing Electrical Consumption in Buildings. *PhD dissertation*, UC Berkeley 2010.

[15] A. Krioukov, G. Fierro, N. Kitaev, and D. Culler. Building application stack (bas). In *Proc. of BuildSys*, pages 72–79. ACM, 2012.

[16] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*, pages 61–66. ACM, 2010.

[17] A. Rowe, M. E. Berges, G. Bhatia, E. Goldman, R. Rajkumar, J. H. Garrett, J. M. Moura, and L. Soibelman. Sensor andrew: Large-scale campus-wide sensing and actuation. *IBM Journal of Research and Development*, 55(1.2):6–1, 2011.

[18] R. P. Singh, S. Keshav, and T. Brecht. A cloud-based consumer-centric architecture for energy data analytics. In *Proceedings of the fourth international conference on Future energy systems*, pages 63–74. ACM, 2013.