Detecting Situations from Heterogeneous Internet of Things Data in Smart City Context

SK Alamgir Hossain[1(✉)], Md. Anisur Rahman[1], and M. Anwar Hossain[2]

[1] Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh
{alamgir,anis}@cseku.ac.bd

[2] Department of Software Engineering CCIS, King Saud University, Riyadh, Kingdom of Saudi Arabia
mahossain@ksu.edu.sa

Abstract. Internet of Things (IoT) offers a lot of benefits for building smart city today and tomorrow. In a smart city, large number of heterogeneous IoT devices are likely to be embedded, which will generate huge volume of data with different formats. Therefore, it is a challenge to address such IoT data heterogeneity and process them together to support decision makers with information of interest. This paper provides a framework to achieve this goal. Our experiment shows that the information obtained from raw IoT data provides situational awareness for the decision makers of smart city in a seamless fashion.

Keywords: Internet of things · Smart city · Situation awareness
Big data analytics

1 Introduction

Today Internet of Things (IoT) has become the cornerstone of smart city development with a push for connected objects, connected devices, and for connected everything. In IoT world, things have the capability to talk to each other and exchange data that contribute to make the city smarter. It is estimated by the International Data Corporation (IDC) that connected IoT devices will surpass 50 billion mark in 2010 [1]. This brings the challenge as how to efficiently process the unprecedented amount of IoT data and use it for human benefits.

Due to the potential of IoT and the popularity it already received, there is an increasing interest in many governments and city officials to provide IoT based public services and management of those services. This realization is called the smart city concept [2]. By collecting different city data, the government may ensure the transparency towards the citizen as well as may offer new services in future [3].

Recently United Nations (UN) stated that over 60% of the world population will live in city environment by 2030 [4]. The more the city population will

K. Arai et al. (Eds.): SAI 2018, AISC 858, pp. 1114–1127, 2019.
https://doi.org/10.1007/978-3-030-01174-1_85

increase the more viable support will be necessary for the citizen. Today, it is difficult to monitor and have full control of a city by depending only on the city officials or law enforcement agencies, who can only collect situation data periodically and take necessary actions passively. On the contrary, in a growing city landscape lots of events will likely occur and any inefficient approach to process and handle those events will hamper timely decision making. To address this challenge in the context of smart city, different projects are already underway that couples IoT with cloud technology to reap the benefit of cloud storage, processing and infrastructure [5,6].

Among different work that leverages the combination of cloud and IoT to provide real-time analytics, authors in [7] put their attention to merge the cloud computing technologies to offer new location aware services as well as reduce the latency. Based on this idea several academic prototype [8,9] as well as commercial services (ThingWorx [10], SmartThings [11]) have emerged today.

At urban scale, SmartSantander[1] introduced a city-wide experimental research where around 20k sensors have been deployed in three different locations. It monitors different activities like park and garden irrigation, parking management, traffic monitoring, weather condition monitoring, etc. By persuading from SmartSantander, Chakrabarty et al. [12] depicted a protected IoT engineering for keen urban communities. They introduced a design containing four essential IoT engineering obstructs that empower a protected Smart City to mitigate digital assaults starting at the IoT hubs themselves. The authors concurred that IoT is in its underlying stage and parcel of research is important.

In the paper [13], the authors investigated the protocols, enabling technologies and architecture of noteworthy urban IoTs. They talked about innovations are near being institutionalized, and industry players are as of now dynamic in the creation of gadgets that exploit these advances to empower the utilizations of intrigue. Actually, while the scope of outline alternatives for IoT frameworks is somewhat wide, the arrangement of open and institutionalized conventions is essentially little. The empowering advancements, moreover, have achieved a level of development that takes into account the reasonable acknowledgment of IoT arrangements and administrations, beginning from field trials that will ideally help clear the vulnerability that still keeps a gigantic reception of the IoT worldview. With the point of conquering the drawbacks of the frameworks said above and enlivened by [13,14] and some important works [12,15], we present an architecture based on IoT for situation awareness in a smart city to help the city administrators.

This paper is composed as follows. Section 2 depicts the situation detection and heterogeneous stream handling mechanism. In Sect. 3 we show the different segments of our proposed framework. Section 4 presents the implementation issues, development challenges of different modules and the outcome that we found from our examination. At the end we provide conclusion of the paper in Sect. 5.

[1] Smartsantander, www.smartsantander.eu.

2 Detecting Situations from Heterogeneous Streams

In a smart city infrastructure, lots of IoT devices with embedded sensors are connected to each other. It is thus imperative for the city administrators to understand the current situation of the city in terms of transportation, healthcare, security and many other aspects based on the IoT data. Efficient situation understanding allows the city officials to provide services to the citizens and perform actions accordingly.

In order to be situation aware, the city officials need to perceive all the events that are generated based on the data from the deployed IoT objects, and reflect a situation. However, this process of being situation awareness lies in the understanding of how the raw IoT data is propagated from the distributed deployment sources to the application level for the decision makers. In the following, Fig. 1 refers to four steps that are part of this data transformation.

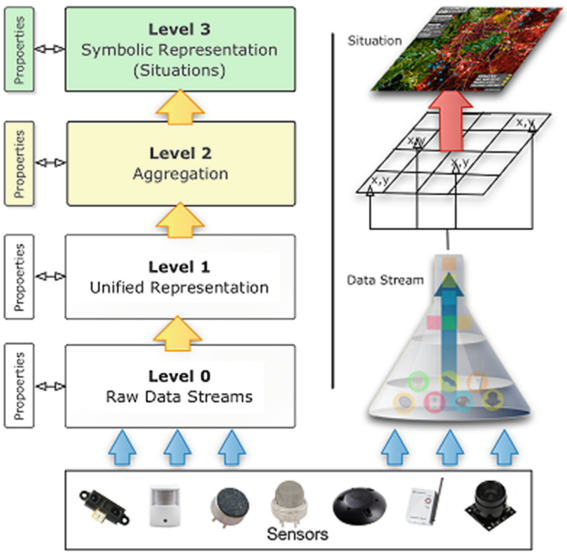


Fig. 1. Proposed situation detection process.

2.1 Raw Data Streams

Without considering the energy consumption if sensor nodes in the IoT devices send their raw data to the server, the battery life will be reduced drastically. For this reason different on-line and off-line mechanism like data compression and aggregation, query modeling are applied to reduce the cost. In our proposed approach, sensor stream is annotated with its location and frequency of creation. Sensor stream is a collection of unbounded data objects (which may have multiple data attributes) in a particular order. The stream may generate in a

specific rate or may vary. The stream may be managed in several steps that are as follows:

(1) IoT Discovery and Search: All the IoT devices that are in the environment must be discoverable so that they can be queried for the data. An efficient look-up mechanisms is necessary but real-world objects is difficult to look-up. So instead of the physical look-up it is better to use a virtual table where the device insertion and removal can be managed dynamically. One of the promising work in this direction is Snoogle [16] which is a search engine of the physical world. It assigns a keyword to sensor and link with associated description of the sensors. Then it stores the keyword with the description in storage with appropriate indexing. Each index point is linked with all the sensors that are connected with it as every sensor in the same area will be assigned the same index point. In this way two-tier hierarchy is used to maintain two type of index point. In the top tier it keeps the key index point and in the bottom tier it maintains all the index point that is associated with the key index point.

(2) Stream Pre-processing: Data pre-processing is an important task in an IoT environment, as data come from noisy environment that require pre-processing before analysis. A real-time pre-processing is necessary in many cases like streaming data processing. Several optimizing technique is useful to make the process faster.

(3) Distributed Processing: For data analytics not only the pre-processing is important but also distributed processing is required due to the existence of numerous devices and the need to collect data from those devices. It is well known that processing multiple data streams in a centralized server is inefficient. So practically for a smart city, decentralized and parallel processing of IoT data is required. But now a question is how to send the data to decentralized processing units without significant loss of performance? We found that ADMM (Alternating Direction Method of Multipliers) [17] provides a technique that is used to solve classical convex optimization problem. In this approach it breaks a problem into some smaller problems so that each smaller task can be handled in a easier way. This model sufficiently matches our case where we have a large set of data that we want to split and process into separate processing units.

(4) Data Storage: After pre-processing, the IoT data need to store in the off-line storage. As the data volume is enormous so the traditional data storage technique is not suitable in case of IoT data processing. According to traditional database management system (DBMS) each record is represent as an object and each column is represented as an attribute. This data storage architecture called a write-optimized system. Instead of using this data model we can store data in a RAW manner or key value manner in which the whole data storage and retrieval will be faster. In our proposed system for the data storage the traditional RDBMS are not suggested for IoT application as the IoT data has no fixed structure and no predefined data model. So in this case NoSQL [18] method of storage technique will be useful. The key idea of NoSQL is storing data in a key-value format without any other data model. Instead of the traditional ACID (Atomicity, Consistency, Isolation, Durability), NoSQL recommends BASE (Basic availability, Soft state, Eventual consistency).

2.2 Unified Representation

In this step the heterogeneous IoT data is converted to a uniform representation, this is important as the huge volume of data need more processing and also difficult to represent. Instead of the original data it is easier to classify the data based on different factors. In our proposed model the data is converted to some unified format such as 'what-when-where' structure. In this approach, a unified information model and communication protocol is necessary. For the unified representation we used the concept W^5HH [19], which is actually a series of questions that extract appropriate result from a problem. This approach is popular in software process and projects management. Here we would like to incorporate the same questionnaire to extract and represent the IoT data.

Table 1. W^5HH principle used to uniform the sensor data

What data is generated	Sensor specific data	Temperature, wind_direction, wind_speed
When data is taken	Date time	10/10/2016 10am, yesterday 1pm
Where the data is coming	Sensor location	Indoor, outdoor, home, office, bed room1
Who is generating the data	Sensor id	1, 2, tempS1, humidityS1
Why (in what condition)	Sensor state/activity	Motion, hot weather etc.
How long data is generating	How long	Ten minutes, one hour
How much resources is needed	Meta data	Data length, data type etc

2.3 Aggregation

In the aggregation step the unified data is converted to a 2-dimensional spatial representation for easy visualization and mapping at the situation level. It combines data that are coming in a particular location and represented in a image like representation we call it S-mage (Situation Image). Each S-mage has several parts that are described in the following sections:

(1) S-Mage Element: Each S-mage element has a collection of *state* and *value* pairs. Here value is the meta-data of the sensor that are extracted from the questionnaire that are listed in Table 1. In a real world environment besides the *state* and *values* the system need to store more information like $latitude \in [-90, 90]$, $longitude \in [-180, 180]$, $altitude \in [0, \infty], time \in [0, \infty]$ etc. with the S-mage element.

(2) S-Mage Set, Π: In the real scenario continuous S-mage will be generated for producing latest situation. We called this finite set of S-mage data to S-mage set, Π.

2.4 Symbolic Representation

Symbolic Representation of Situation is a set of information of an environment over a period of time. In the final step the 2-dimensional spatial S-mage data is converted to appropriate situation. The process of symbolic representation is a classification task where each cell of the 2-dimensional representation represents a particular event.

3 Proposed Architecture

In this section we present different components of our proposed architecture and their functional descriptions. The components of the system are depicted in Fig. 2 as a block diagram. The architecture has three main components: Device Module, IoT Cloud Middleware and Subscribers. The device module includes sensors and other objects that may sense the environment. In reality things that can produce time dependent data series may be considered as a sensor. IoT Cloud Middleware is responsible to publish the message and subscribe the clients to receive the messages. This module is also responsible for message routing. Finally the clients or subscribers receive the services from the system. In our case the display devices where the situation for the city will be presented are consider as the subscribers. Generally any object that want to receive data from the IoT cloud platform may be considered as a client. It is also possible for an object to be both a Sensor and Client in our architecture. We provide the functional description of each of the three components in the following sections:

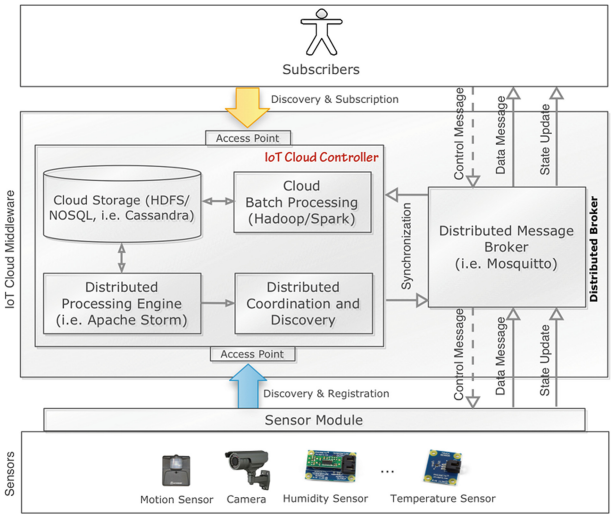


Fig. 2. A bird s-eye view of our proposed architecture.

3.1 Device Module

This module is responsible for all the physical linkage between the IoT device and the IoT cloud platform. Usually by using different standard communication technology like NFC (Near Field Communication), GSM, WLAN, GPS and sensor networks together with SIM-card technology etc. are used to connect the device module to the physical sensors. For every connection mechanism it is important to save the bandwidth. The data from the sensors to the IoT platform may be transferred either asynchronously or synchronously. The main functionality of the device module is to send data but these modules also sense control message that are generated by the IoT middleware. The control message is important in many cases to control the sensors. For example zoom, pan, tilt and resolution functionality control of a camera sensor, precision, accuracy, sensitivity, range, calibration control of temperature, humidity, accelerometer, IR, distance sensors.

3.2 IoT Cloud Middleware

In our proposed architecture we introduced a cloud based IoT middleware that controls and processes IoT data. It is an important question whether the data processing is performed in a centralized server or in a distributed manner. In case of centralized processing we need a super power machine which is not feasible, so our next choice is the distributed processing. Using this type of processing the information may be processed in the intelligent devices like smart phone, sensors and finally process in the distributed nodes. As the terminal devices have limited computational capability, most of the task is processed in the cloud nodes. This set of cloud nodes act as a cloud middle-ware, which has the following two sub-modules.

(1) IoT Cloud Controller: The main functionality of this module is to coordinate synchronized communication with other modules as well as to perform system management services. The controller sends or receives messages from the Distributed Message Broker to create message route between subscribers and IoT devices. The controller also maintains a repository of IoT data and different meta-data information that are necessary to discover, filter and manage the different services. In our architecture we prefer to use Apache Cassandra [20] for storage management. Our preference on Cassandra over other NoSQL database (like MongoDB) or other HDFS based cloud storage is due to its ability to scale while still being reliable. It is possible to deploy Cassandra across multiple servers without lots of overhead. A distributed processing engine is responsible to process unbounded streams of data specially in case of real-time processing. Finally, as the IoT devices are developed by different manufactures without following a common standard, interoperability becomes a big issue. In our architecture we achieved interoperability between the devices and the platform by using SOAP based on Web Services [21].

(2) Distributed Message Broker: Generally a message broker is a program that can receive message, translate the message and finally can send message. As we already discussed in earlier section that pub/sub based message delivery is the most popular in IoT scenario, we used MQTT [22] based pub/sub distributed message delivery mechanism for asynchronous messaging.

3.3 Subscribers

Subscribers are those who want to consume service, which can be a user or a device or any other source that are sending the message. In the architecture, clients can discover and subscribe to the system through the cloud controller. It should be noted here that a single software program may have both sensor and client role in the system. For example a face detection subscriber may produce face video data as well as may look face image inside the video through appropriate subscription to the system.

4 Implementation and Result

This section describes the implementation details of the prototype system that reflects the architecture presented in Sect. 3.

4.1 Experimental Setup

In our experiment we setup a set of sensors as listed in Table 2 on some predefined locations in our city. We selected the location in a manner that we can freely access the environment. The area is approximately $0.45\,\mathrm{km}^2$. Figure 3 shows a map view of the deployment. In this map view different pin in the map is representing the different nodes. We installed nodes to detect light level, environment noise level, temperature of the environment, traffic presence specially car presence. After selecting the test bed location our main challenge to place the sensor nodes in appropriate locations. The main challenge was to provide uninterrupted power to the sensors nodes. Practically it is more convenient to use solar energy for battery charges; we used small solar panel to charge the node batteries. Solar power has the main problem the battery power may be finished during the night but we placed our nodes in the road side lamppost (as illustrated by the picture in Fig. 4a) which is powered at night by the municipality authority. Some of our node needs more power we connected those node in the lamppost so that it is can be functional at night.

We developed a multi-modal sensor node as shown in Fig. 4b. Each sensor node contain camera, distance, motion, and temperature sensor. Finally we used a metal box to protect the sensor node from weather or any human.

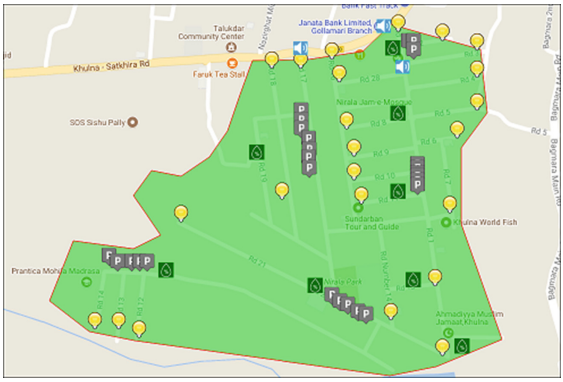


Fig. 3. Prototype deployment map view.

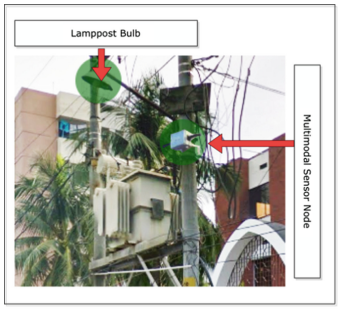
Table 2. Node and sensors used in the prototype

Node type	Amount	Sensors
Humidity	08	Humidity
Light	23	Light, temperature, acceleration
Noise	03	Noise, acceleration
Parking	26	Occupancy
Smart phone	15	Smart phone sensors
TOTAL	**75 Nodes**	**160+ Sensors**

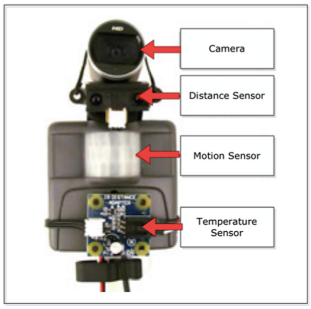
In our system we used mobile phone as sensors to sense different environmental data like noise, temperature and several physical data for example GPS coordinates. If any specific event is currently occurring in the city the system will alert the city officials that are subscribed to that service. User also can report the occurrence of such events and it will automatically be propagated to other subscribers of the respective types of events.

(1) Infrastructure and Database Management: In our prototype we used OpenStack[2] to create a standard and scalable complaint cloud infrastructure. This open source cloud operating system enables to create and offer cloud computing services running on specific standard.

[2] OpenStack, http://www.openstack.org.



(a) Sensor node in the lamppost.



(b) Multi-modal sensor node.

Fig. 4. Sensor deployment.

4.2 Result

In order to evaluate the performance of our prototype we conducted several measurement studies. All those studies can be used to justify the acceptability of our approach. At first we test the processing time of our system and then we test the situation detection performance (Fig. 5).

(1) Processing Time: In order to measure how our system is performing in terms of time we embedded monitoring hooks in the system and recorded the responses of those. Finally we analysis the process time from the sensor node to the online user (city official) by using the following (1). In this equation, α is the processing time of the cloud platform, I is the sensor's average sense time from the environment and sensor processing time, Π is the average data transmission rate from sensor to the cloud platform, β is the time required to send from cloud platform to the user devices or sensors and finally n is the message size.

$$R = I + \frac{n}{\Pi} + \alpha + \beta \tag{1}$$

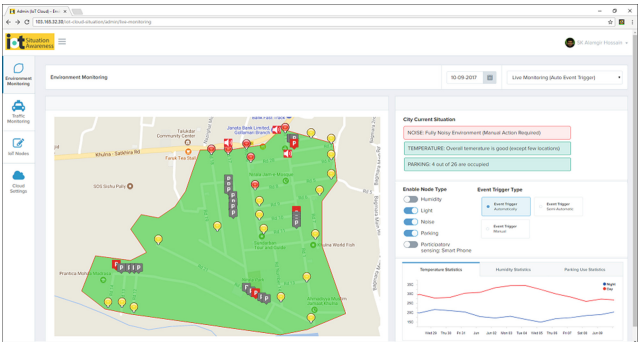


Fig. 5. Prototype web application interface for live situation monitoring.

After any event occur in the environment the system approximately requires $R = (352 + 266 + 4810 + 360)$ ms to finish the task. In our test the sensor node processing time is 352 ms, network delay is 266 ms and α and β is 4810 ms and 360 ms, respectively. In Fig. 6 the performance of the sensor and cloud platform modules is depicted.

(2) Situation Detection: In order to test the situation detection performance we monitor the environment approximately 4 weeks and calculated the number of event occurred versus the event detected by our method. Out of approximately 181 different events like high temperature, low humidity, hot sunny day our prototype able to detect 176 events perfectly. Sometimes our algorithm shows misleading information due to the communication delay as well as the server load. It should be noted here that the entire test performed during summer days and there was no situation like rain or other natural disasters.

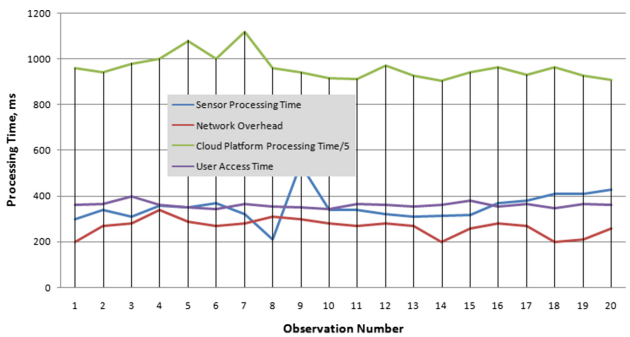


Fig. 6. Processing time of different modules.

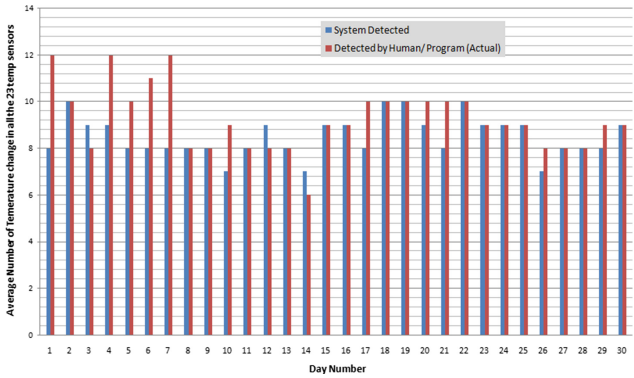


Fig. 7. Situation detection performance on 23 temperature sensor over a 4 weeks.

In Fig. 7 demonstrates the situation detection performance over temperature sensors that we used in our prototype test case. There are total 5506 temperature raise or drop during the measured days. As we performed our test during sunny day so we consider the threshold temperature as 25 °C. So a separate sensor is placed in the same location where the nodes are located to calculate the actual temperature raise or fall from the threshold. From this analysis we found that our cloud system produced almost identical result as the actual data. It should be noted here that sometimes our system shows more raise or drop than the actual because sometime for a continuous run or direct sunlight the temperature sensor shows out performance.

5 Conclusion

In a smart city, it is a challenging task to process and detect the situation from the huge volume of data that will be generated from large number of heterogeneous IoT devices. In this paper we presented a framework for detecting situations from heterogeneous Internet of Things data in a smart city context. Our experiment shows that the information obtained from raw IoT data provides situational awareness for the decision makers of smart city in a seamless fashion. The research work is our ongoing work and we are trying to implement the proposed architecture in a more resource constant and complex scenario where the devices will be connected and disconnected like the natural manner as well as the scalability issues will be considered. Still we believe the research work that we described in this paper will be helpful in the society as well as the research community.

References

1. I.D.C. (IDC): Worldwide smart connected device shipments. Technical report, March (2012). http://www.idc.com/getdoc.jsp?containerId=prUS23398412
2. Schaffers, H., Komninos, N., Pallot, M., Trousse, B., Nilsson, M., Oliveira, A.: Smart cities and the future internet: towards cooperation frameworks for open innovation. In: The Future Internet Assembly, pp. 431–446. Springer (2011)
3. Cuff, D., Hansen, M., Kang, J.: Urban sensing: out of the woods. Commun. ACM **51**(3), 24–33 (2008)
4. Naphade, M., Banavar, G., Harrison, C., Paraszczak, J., Morris, R.: Smarter cities and their innovation challenges. Computer **44**(6), 32–39 (2011)
5. Vanelli, B., et al.: Internet of things data storage infrastructure in the cloud using NoSQL databases. IEEE Lat. Am. Trans. **15**(4), 737–743 (2017)
6. Ab Rahman, N.H., Cahyani, N.D.W., Choo, K.-K.R.: Cloud incident handling and forensic-by-design: cloud storage as a case study. Concurr. Comput. Pract. Exp. **29**(14), e3868 (2017)
7. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile cloud computing, pp. 13–16. ACM (2012)
8. Fox, G.C., Kamburugamuve, S., Hartman, R.D.: Architecture and measured characteristics of a cloud based internet of things. In: 2012 International Conference on Collaboration Technologies and Systems (CTS), pp. 6–12. IEEE (2012)
9. Li, F., Vögler, M., Claeßens, M., Dustdar, S.: Efficient and scalable IoT service delivery on cloud. In: IEEE CLOUD, pp. 740–747 (2013)
10. ThingWorx.: Internet of things and M2M application platform. PTC, Technical report (2016). http://www.thingworx.com
11. SmartThings.: Smartthings open cloud. Samsung, Technical report (2016). https://www.smartthings.com/opencloud
12. Chakrabarty, S., Engels, D.W.: A secure IoT architecture for smart cities. In: 13th IEEE Annual Consumer Communications Networking Conference (CCNC), pp. 812–813. Jan (2016)
13. Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of things for smart cities. IEEE Internet Things J. **1**(1), 22–32 (2014)
14. Sanchez, L., et al.: Smartsantander: the meeting point between future internet research and experimentation and the smart cities. In: Future Network & Mobile Summit (FutureNetw) 2011, pp. 1–8. IEEE (2011)
15. Sakhardande, P., Hanagal, S., Kulkarni, S.: Design of disaster management system using IoT based interconnected network with smart city monitoring. In: 2016 International Conference on Internet of Things and Applications (IOTA), pp. 185–190. Jan (2016)
16. Wang, H., Tan, C.C., Li, Q.: Snoogle: a search engine for the physical world. In: The 27th Conference on Computer Communications (INFOCOM), pp. 1382–1390. IEEE (2008)
17. Boyd, S.: Alternating direction method of multipliers. In: Talk at NIPS Workshop on Optimization and Machine Learning (2011)
18. Fan, T., Chen, Y.: A scheme of data management in the internet of things. In: 2nd IEEE International Conference on Network Infrastructure and Digital Content, pp. 110–114. IEEE (2010)
19. Boehm, B.: Anchoring the software process. IEEE Softw. **13**(4), 73–82 (1996)
20. Gupta, A.: A big data for apache cassandra. J. Web Dev. Web Des. **1**, 1–3 (2017)

21. Paik, H.-y., Lemos, A.L., Barukh, M.C., Benatallah, B., Natarajan, A.: Web services–soap and WSDL. In: Web Service Implementation and Composition Techniques, pp. 25–66. Springer (2017)
22. OASIS.: Mqtt 3.1.1 specification. Technical report. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html