

Smart City Data Streaming and Analytics

Shreyas Ramakrishna, Veena Nalluri, Sanchita Basak and Anabil Munshi
April 17, 2018

Abstract—Increase in IoT and CPS devices, has given way to increased attempts on building smart cities equipped with thousands of sensors. With large number of sensors, there will be huge amounts of data being generated and streamed. However easy and interesting the idea may sound, there is a requirement to build a robust platform to assimilate all the data being collected and also provide a meaning to the data being collected. This work proposes a novel architecture for data streaming and analytics in smart cities. The architectural idea discussed here uses some distributed concepts of streaming, analytics along with various computing paradigms of cloud, community cloud and fog computing.

keywords: Distributed streaming, Distributed analytics, Computing paradigms, Data Management

I. INTRODUCTION

A. Smart City: Overview

As a direct effect of urbanization all over the world, a major chunk of the global population has started to migrate into cities. WHO estimates that, by 2050, 66% of the world's population will be living in urban areas. With this influx of people in cities and towns, providing efficient mobility, energy management, power supply and other essential services to urban populations becomes a daunting task, because everyone is competing for resources that are being constantly consumed at a very high rate.

This necessitates the development of "smart" cities which use efficient networking and data storage infrastructure to handle these issues.

While researchers have their differences in what constitutes a "smart city", an overarching definition which most seem to agree upon is that - "A smart city is an urban area that uses different types of electronic data collection sensors to supply information which is used to manage assets and resources efficiently." [Ref] The sensors may be automated or may also collect crowdsourced data through mobile applications.

The possibilities for realization of these smart urban centers have been made more pronounced in recent years by the growth in information and communication technology (ICT). The concept of a "smart city" integrates ICT and internet-enabled devices (Internet of Things) to provide better performance, and fault prediction, reduce resource consumption, increase energy efficiency and provide an overall better quality of life to citizens.

B. The smart city loop

Schleider et al., in their 2015 paper which discusses a roadmap for building next-gen smart cities [8], present an

ideal smart city loop (shown in Fig. 1). The ideal smart city loop, as presented in [8], consists of four sequential recursive sections:

(a) Data from city: The city emits various amounts of data, both static and dynamic, from different sources including IoT networks and citizens.

(b) Management of diversified data about the city: This calls for the data to be properly managed and integrated (ying together services, mobility, infrastructure and energy) for modeling purposes.

(c) Multi domain expert network providing analytics and modeling: Models of different aspects of essential city services are then made, to gain a holistic view of the functioning of these services.

(d) Decision-making support to government and citizens: Analyzing these models and reporting analytics in real-time gives the decision support that is required to address complex challenges like network efficiency, environmental sustainability, prediction of future conditions, etc.

This loop provides a baseline for realizing a smart city. Therefore, in this paper, we try to leverage cloud-based networking framework for collecting smart city data from various sensors and thereby integrate this data to create a model and analytics platform for city services.

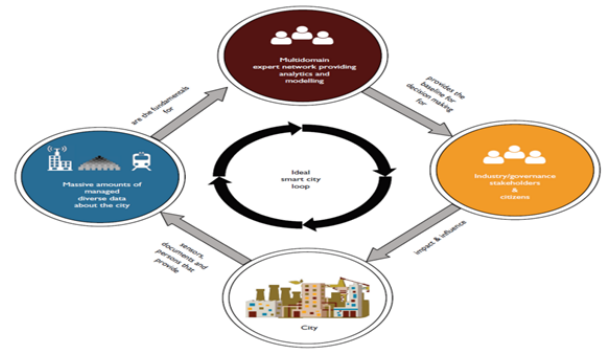


Fig. 1. The Ideal Smart City Loop (Schleider et al., 2015)

C. Background: Ongoing smart city projects

Researchers are undertaking several projects in different parts of the world to build sustainable "smart" urban centers. Some of the notable ones are mentioned below:

(a) Sterling Ranch: This is a 12,000-home smart city project, currently under development in Colorado, USA. Researchers from Vanderbilt University, Nashville are

testing out a number of technologies like water quality monitoring, rainwater harvesting, demand-and-supply electricity management techniques (smart grid) and solar power functionalities in this facility. The entire city is connected by a fiber-optic network. Once fully developed, residents will be able to monitor their water and energy usage.

(b) Array of Things: This is a real-time networked urban sensor project in Chicago, USA. The developers describe this as serving the purpose of a fitness tracker for the city. In this project, sensor boxes or "nodes" are being placed at various locations in the city of Chicago, attached to traffic light poles, to collect real-time data on the city's environment and activity for research into how infrastructure can be improved following the smart city structure. These sensors collect localized information on temperature, air quality, pedestrian and vehicular traffic of locations within the city, with the goal of providing the citizens with real-time block-by-block updates of city services and environment.

(c) Fujisawa SST: This is a sustainable smart town in Fujisawa, Japan. The town provides total mobility services offering ridesharing facilities and linking electric vehicles and EV bikes. The town is connected by town energy networks and town information networks to collect data on citizens' energy use so that they can be provided with personalized energy-saving recommendations that are tailored to their individual way of life, while ensuring data privacy.

All of these projects are exemplary in showing how efficient architectural and networking implementations can lead to the development of truly "smart" urban ecosystems.

We now move on to discuss the available literature on smart city architectural principles that we used for this paper.

II. RELATED WORK

This section has some of the work which we reviewed to build our novel smart city architecture.

A. Dataflow Architecture

Benson et al.[2] proposed Safe Community Awareness and Alerting Network (SCALE), a cyber-physical system (CPS) leveraging the pervasive Internet of Things (IoT) and used this multi network architecture to build a smarter, safer home to all residents at a low incremental cost. It comprises of various sensor devices, networking protocols, cloud platforms and an efficient middleware to sense, analyze the events sensed and provide feedback to the residents or users.

In order to enhance machine-to-machine (m2m) communication for exchanging IoT data in SCALE (sensed events, analytics, alerts, etc.), the authors propose the Data in Motion Exchange (DIME) system, a communication hub for IoT simplifying the development and deployment processes. The architecture proposed by the authors are shown below.

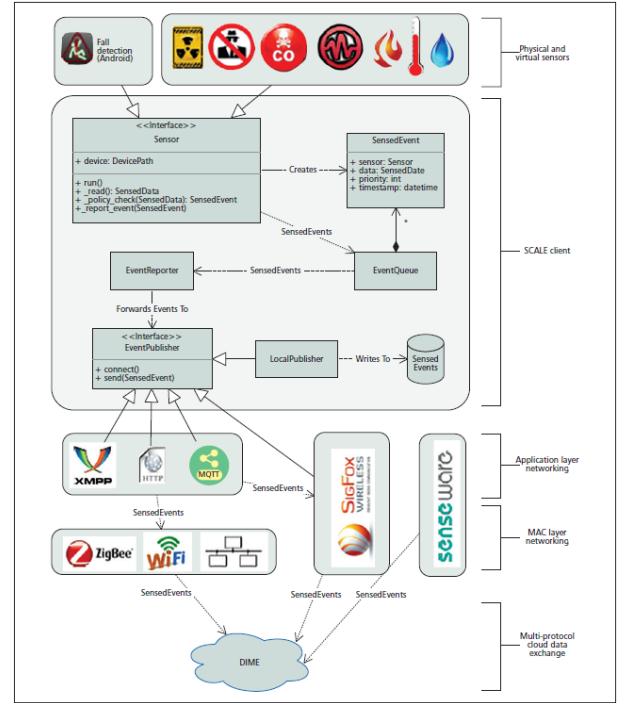


Fig. 2. The SCALE architecture

So, we see that SCALE provides an architecture on how to collect sensor data and store it in the cloud. Therefore SCALE acts as an event-driven middle-ware platform where sensor devices can upload sensed events to a cloud data exchange and the analytics service retrieves these events, scan for possible anomalies and send residents the corresponding feedback.

In SCALE2, Uddin et al.,[10] have discussed an architecture that allows independent development and deployment of applications. The architecture provides a clear delineation between application logic and underlying device protocols, allowing for a clear path to support present standards-based protocol implementations as well as to extend support for future implementations without constraining the application to change in common use cases. So, the platform is built such that each application task runs independently within the user space.

B. Data Streaming

In figure 2, we see that there are several physical sensors sensing intensity of light, smoke, temperature, humidity etc. Similar to this, in our implementation we have a multisensor sensing the events along with the time-stamp and the sensor is communicating with a Z-stick through open Z-wave protocol which is an open source C++ library to control the Z-wave networks via a USB Z-wave controller.

So, in the architecture proposed by the authors in SCALE2, they have used MQTT or http to store the sensed events in the cloud. In our implementation which will be described in detail in the next section, the events sensed through Z-stick is written in local InfluxDB and the communication between local InfluxDB and global InfluxDB which

is running on an instance of cloud is done through Zmq publish-subscribe pattern. Thus the data is getting stored in cloud.

C. Data Management

Backend data management for queries and analytics has become vital with the increase in IoT data. Pecan street smart grid data management project performed in Austin, Texas is one of the first architectures for data query and analytics. In this project, highly granular data with 15-second resolution on resource generation and consumption, including total consumption of electricity, water, and natural gas and solar generation, are collected for more than 100 homes.

Measuring home energy use data is significant for reducing energy consumption and impact on the environment. Currently, given the development of information and communication technology, data are easily collected over ubiquitous integrated networks such as the Internet. The Pecan Street smart grid demonstration project in Austin, Texas is one-of-a-kind project due to its customer-focused approach. This project seeks to show the utility of the smart grid to the homeowner due to the highly visible roll out of integrated end-user systems such as home energy management systems (HEMS) that can manage the energy usage of numerous consumer systems including appliances, consumer electronics, and electric vehicles.

Fig 2 shows the data access sequence. The user will connect to the database giving proper access information through MySQL. Once the connection is done, the user will write queries and execute them on database.

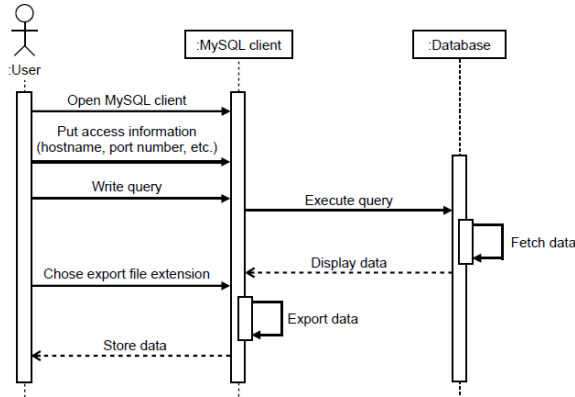


Fig. 3. A data access sequence diagram through using MySQL client

Fig 3 illustrates the activity diagram of automatic graph generation. This activity diagram consists of three processes: preparation, data extraction, and graph generation. In the preparation process, the user will export the DataIDs using the MySQL client. In the data extraction process, the Java program initially reads the DataIDs and stores them in an array list for a looping condition. Each DataID is read line by line and input into a query statement routine until the list of the DataIDs becomes null. At the end of every routine, the program exports the extracted data file. After finishing the

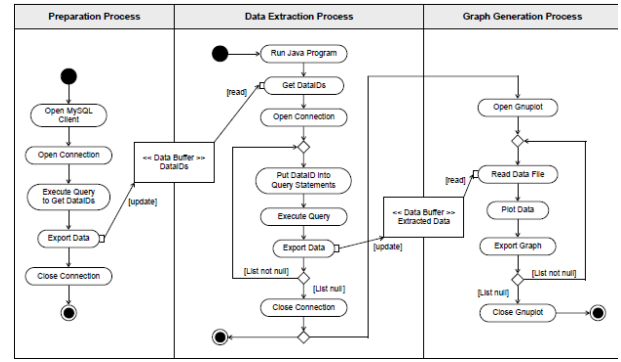


Fig. 4. An activity diagram of automatic graph generation

routine, the program opens GnuPlot. This graph generation process also runs the same routine and generates graphs.

D. Data Analytics

This section discusses on different data processing technologies. There are 3 types of data processing techniques: Batch processing, Stream processing and Complex event processing.

(a) **Batch Processing** : Batch processing is where the processing happens of blocks of data that have already been stored over a period of time. For example processing all the transaction that have been performed by a major financial firm in a week. This data contains millions of records for a day that can be stored as a file or record etc. This particular file will undergo processing at the end of the day for various analysis that firm wants to do. Obviously it will take large amount of time for that file to be processed. Batch processing works well in situations where you dont need real-time analytics results, and when it is more important to process large volumes of information than it is to get fast analytics results.

(b) **Stream Processing** : Stream processing is a golden key if you want analytics results in real time. By building data streams, you can feed data into analytics tools as soon as it is generated and get instant analytics results using platforms like Spark Streaming. Apache Storm is a stream processing framework.

(c) **Complex Event Processing** : Complex event processing (CEP) is one of the key technologies for supporting real-time analytics. It processes data as they are flowing, without writing them to the disk and supports complex temporal queries. System computes over multiple streams of data as it enters the system. CEP use case may include large number of events, queries or complex queries and a single CEP node will not be sufficient to handle those scenarios, hence it is essential to scale CEP query execution so that they can run across several machines and handle much larger event rates. Wihidium discusses how to balance workload among nodes efficiently and explains how distributed CEP can be performed on large amounts

of data. The paper focuses on three technologies used for scaling queries: Pipelining, Partitioning and distributed operators.

E. Computing paradigm

With the increase in the use of sensors and IOT, huge amount of data started being generated, but then the question that came into frame was about storing this huge data. It was not possible to store the data on physical devices of data, as it was not sufficient. The figure below shows the increase in the IOT data over the years.

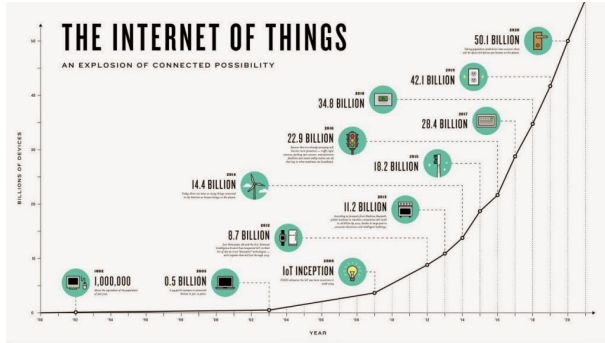


Fig. 5. Increase in IoT data

As there was an exponential increase in the IoT data, different computing paradigms came into existence. Some of the existing computing paradigms are cloud computing, Mobile cloud computing, cloudlet, fog computing, edge computing, mobile edge computing, mist computing, social computing and dispersed computing.

Out of these wide variety of platforms cloud computing was one of the first to be introduced and is also the most popular and widely used.

1) **Cloud Computing (CC)**: Is a service model where computing services that are available remotely allow users to access applications, data, and physical computation resources over a network, on demand, through access devices that can be highly heterogeneous. As in [5] a cloud infrastructure comprises of a physical layer consisting of the all the hardware components like server and storage components required for the cloud and an abstraction layer consisting of the software to be deployed over the physical layer. In cloud computing [4], resources are rented in an on demand and pay-per-use fashion from cloud providers. Just as a huge hardware machine, cloud computing data centers deliver an infrastructure, platform, and software applications as services that are available to consumers. This facilitates offloading of highly consuming tasks to cloud servers. The National institute for standard and technology (NIST) is responsible for providing standards and guidelines for providing security to all assets. The work [5] provides an insight into the cloud computing infrastructure which consists of three service models, four deployment models and five essential characteristics of cloud computing which are; on-demand self-service, broad network access, resource

pooling, rapid elasticity and measured service. A cloud service model represents the package of IT resources required by the consumers as service and provided by the cloud vendor. The three cloud service models are: Software as a service (SaaS): The consumers are only provided with the capability to run the applications of the provider, but they have no control over the cloud infrastructures like operating system, servers or storage. Platform as a service (PaaS): The consumers have the capability to deploy either own or acquired applications to the cloud. The consumer does not have any control on the cloud infrastructure, but has control over the deployed application. Infrastructure as a service (IaaS): The consumers can use the applications provided on the cloud without the necessity to download the application to the consumers computer. The classification of cloud computing models based on their deployment ways are; private cloud, public cloud, community cloud and hybrid cloud. The literature discussed here provides a basic understanding to cloud computing along with the skeleton of cloud computing infrastructure. There are cloud vendors like Amazon, Microsoft, Google which provides services of public, private and hybrid clouds. (a) Pros: Flexible capacity, availability, accessible, reliability, availability. (b) Cons: Privacy, security, cost and latency for safety critical systems.

2) **Multi-tier cloud**: In [6], the authors introduce MAP-Cloud a hybrid tiered cloud architecture consisting of local and public clouds, which could be leveraged to increase both the performance and scalability of mobile applications. This work utilizes some real-time applications and their computational requirements along with the QoS of power, delay and latency to move from a 1-tier cloud to a 2-tier cloud. This work is based on the fundamental concept that, the QoS improves as the computation is performed closer to the user device. The primary cloud in this architecture could be the services of the cloud vendor and the second tier can be simple servers placed close to the mobile devices.

The other interesting contribution the paper makes is Resource allocation, to distribute the computation among the two tiers of the cloud. There have been a whole slew of papers which revolves around the resource allocation problem. So, this paper along with results convinces the use of multi-tier cloud system to perform computation close to the source in order to improve the QoS and reduce power consumption.

3) **Community Cloud**: Cloud Computing has always provided a compelling value proposition for organizations to outsource their data for storage and analytics. However, there are growing concerns over the monopoly of large Cloud vendors, including the lack of information privacy, security and cost issues. Also, the data centers required for Cloud Computing are growing exponentially, creating an ever-increasing carbon footprint, raising environmental concerns.

The social paradigms and technologies of Digital Ecosys-

tems, including the community ownership of digital infrastructure, can provide a solution and help us overcome concerns. So, Cloud Computing combined with the principles of Digital Ecosystems provides a compelling socio-technical conceptualization for sustainable distributed computing.

Spare public resources of networked phones and computers can be used as virtual data center to form a Community Cloud as described in [3]. However interesting and simple this idea of community cloud may look, but the idea is still naive and there are many unanswered questions like:

- (a) Core infrastructure: Challenges the likes of building a fully functional peer-to-peer network which would spin up a cloud.
- (b) Identity/ Trust: Is trusting of an unknown node in the p2p network possible, is one question which has to be thought of.
- (c) Distributed computing: Distributing incoming job among various available resources for computation.
- (d) Bandwidth management: Satisfying the increase in the band width requirements of the community cloud.
- (e) Community currency: What would be the mode of payment for using a node's resource in the p2p network. Should it be for free or should we think of something like community currency.
- (g) Remote Service Execution: When a service is needed to fulfill a request, but is not currently instantiated on a suitable node, a copy should be retrieved from the repository and instantiated as needed.

4) **Edge Computing (EC):** Is similar to the fog computing and the cloudlet concept except for a few key differences. Typically, edge computing machines are low powered devices and are not shared between different applications. Also in EC the computation is completely localized to the edge nodes however in fog computing (FC) the computation and communication is shifted towards the edge of the network.

The disadvantage of CC and MCC of high network latency and the growth of IoT has given rise to the idea of moving the cloud or computation closer to the devices and this lead to the genesis of edge computing paradigm. Idea behind EC is to perform computation and storage locally within the resources available at the edge devices. As in [9] tends to address the potential issues of response time requirements, battery life constraints, data security and privacy, and bandwidth reduction. It also discusses the evolution of the edge computing from the concepts of CC and IoT, provides a vision of EC definition and several case studies to support the idea of edge computing and the inherent advantages it has to offer.

5) **Fog Computing (FC):** It was introduced to solve the problem of having billions of IoT devices deployed over time that cannot operate by simply having connectivity to servers in the cloud; instead, computations are pushed closer to them (to the leaf nodes). Unlike the traditional Internet of Things model, the fog computing model, pioneered by the Open Fog

Consortium suggests the use of shared computation servers, similar to the vision of cloudlet described by [7]. However, the key difference lies in the software as a service pioneered by fog computing. For example, instead of just providing the computation resources, a fog computing machine often provides machine learning stack as a service. Also, a difference with respect to cloud computing is that fog computing supports user mobility. Nevertheless, fog and cloud are not independent paradigms as in a fog computing environment there is the need for interacting with the cloud to achieve coherent data management.

As mentioned in [11] the unresolved issues in cloud computing of latency and mobility have been overcome by providing services and elastic resources at the edge of the network. The literature provides an overview of fog computing by providing a definition of fog computing and discussing the issues related to fog computing like fog networking, Quality of Service (QoS), interfacing and programming model, computation offloading, accounting, billing and monitoring, provisioning and resource management, and security and privacy. Along with providing insights into the issues related to fog computing, it also mentions the applications like augmented reality (AR) and Real-time video analytics, content delivery and caching, and mobile big data analytics which will promote fog computing.

FC provides various advantages of low latency, location awareness, real time applications, heterogeneity, and end device mobility which make it an attractive computation paradigm. But, the security and privacy challenges of trust, authentication and man-in-the-middle attack discussed above makes it challenging to implement the FCN in daily life applications.

III. ARCHITECTURAL OVERVIEW

From all the material we discussed in the literature survey, we propose our novel architecture which is an assimilation of multiple ideas in the fields of computing paradigm, data streaming, data analytics and database management. In the context of a smart city all of these areas play an important role. So, every single department has to be paid attention and weighed equally. Some of these department and its prominence are discussed below:

Data Streaming: As we have discussed before, huge amount of sensor data will be generated in a smart community and will also be streamed to a storage location. So streaming plays a very important role in transporting data between the source and the data collector (database). From our literature survey we found a number of streaming solutions like ZMQ, MQTT and KAFKA are available for streaming the data between the source and destination. Data streaming plays a vital role when real time data processing is involved, so obviously in a smart city where real time data processing is vital, streaming of data plays a key role.

Data Analytics: We know that huge amount of data will be collected from the smart city, and data analytics deals with making useful meaning of the data being collected in the database. As discussed in the previous sections, there are two different types of data processing namely batch processing and stream processing. In our scenario, we will mainly deal with stream processing, which means we will be processing stream of data and do not wait for batch of data to be collected. However, a special case of stream processing called Complex Event Processing (CEP), has been frequently used with IOT data. It is very similar to data streaming, but has multiple streams of data which will be processed together. We plan to work with tools of Spark or WSO2 Siddhi for our data analytic end.

Computing paradigm: We have discussed about the various interesting computing paradigms of cloud computing and fog computing in our previous sections. Our smart city architecture accommodates processing at the cloud, community cloud and fog computing.

Database Management: Once we have decided with the different computing paradigms, we will have to have instances of our databases running in their servers. Some available databases we had a look at was cassandra, MongoDB and InfluxDB. Since InfluxDB is a realtime series database, it was our natural choice to be used. InfluxDB also has a simple InfluxQL to perform queries/ analytics on the data.

Along with the realtime series, opensource and ease of use advantage, it also has an inbuilt support to grafana, which is an open platform tools and analytics. This could be used to provide some interactive feedback to the users about different sensed values like water consumption, electricity usage, etc.

These four form the basic building blocks for our architecture over which we could be using different sensors to obtain some real values from the daily lives of a city. Some of the important sensor values we are planning to track here include:

- (a) Smart Home sensor data: Includes water consumption, electricity usage, temperature, humidity, ultraviolet and motion sensor readings.
- (b) Air quality monitoring data: Includes readings of chemicals like carbon-di-oxide, carbon-monoxide, benzene, trichloroethane, etc.

The figure below shows the architectural overview of the smart city which includes smart homes equipped with multiple sensors, around 20 to be precise with. It also includes multiple air quality sensors equipped on streets to read different chemical values of the surrounding area. Throughout the whole work we consider a large scale smart city which may have homes somewhere in between 2000 homes to 12000 homes.

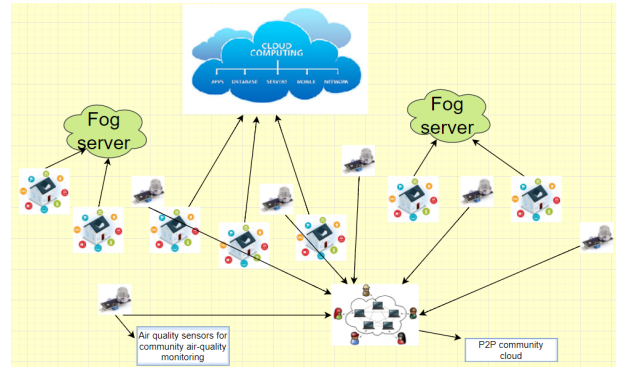


Fig. 6. Different actors in the smart city

A. Air quality monitoring

Air quality monitoring plays a very important role in the modern day community, where people are health conscious and want to know the kind of air quality they are breathing. There have been works before which aimed at monitoring the air quality of the surrounding. In[4], the authors have provided an architecture to stream the air quality data collected from sensor and then stream it over 3g data to a database. Similar to this work, there has also been a realtime project called Array of Things, briefly discussed in the ongoing project session. So, there has been a lot of prominence for monitoring the air quality, and thus we have multiple air quality sensors included for in our architecture.

B. Smart Homes

Our architecture for sensing home data and streaming data has been inspired from the real smart home community called sterling Ranch in Colorado. These smart homes have around 20 sensors equipped in different locations, which measure different readings discussed above. The figure below shows a schematic of the complete home data collection, streaming, management and analytics. The implementation below explains in detail the different actors involved in the overall data streaming and analytics of smart homes.

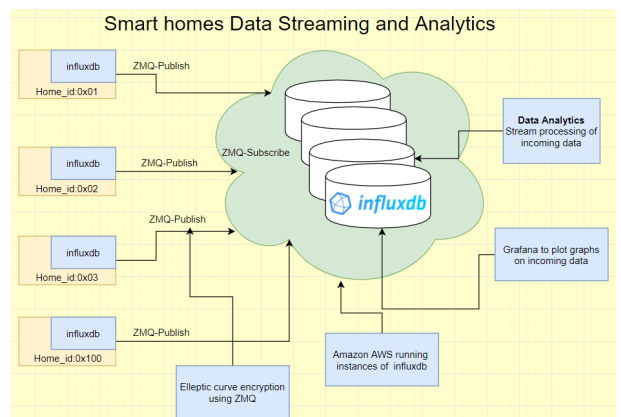


Fig. 7. Different actors in the smart home

IV. IMPLEMENTATION

This section talks in depth about our implementation. We describe exactly with figures about the novel architecture we have proposed. We have explained with figures various important actors and also the tools we have used to implement our architecture. As represented in the figure2, we have worked on our basic architectural requirements to build an architecture for smart home data streaming and analytics. The figure2 provides information about all the different actors involved, about our data streaming tool, data management and also about our future idea of building a back end data analytic running on the data received in the database. In the remaining part of this section we will explain about the different important actors (building blocks) that we have used in our implementation.

A. Data Collection

This section basically talks about the sensing part of our architecture. We have used sensors which works with Openzwave libraries. OpenZWave is an open-source, cross-platform library designed to enable anyone to add support for Z-Wave home-automation devices to their applications, without requiring any in depth knowledge of the Z-Wave protocol [1]. It's range is about 30m and has been widely used in home automation devices, for short range communication.

In our scenario, we have equipped homes with multiple sensors working with OpenZwave library which then communicates with a z-wave controller (aka Z-stick). The Z-stick acts as the controller to start the z-wave network and then adds the sensors into the network. So, the controller is responsible for querying the sensors and obtaining the updates from the sensors.

Along with the devices using OpenZwave library, this work has also used an home automation device called Homeseer, which is a home controller based on linux. This controller along with the Z-stick forms the data collection block. The next parts of this section will explain the remaining important actors of our architecture.

B. Data streaming

Once the data from sensors is collected in Z-stick, it is transmitted to local influx dB using DB write operation. Each home will have a local influx DB, all the sensor data will be transmitted each time there is any change in data values or if there is no change in data every hour the data is transmitted to Z-stick and Z-stick writes the data into Local influx dB. Several instances of influx dB are maintained in the Amazon AWS and data transmission from local influx dB placed at home to global influx dB is done through ZMQ Publish Subscribe. The local instance of influx dB acts as a publisher and global instance of influx dB acts as a subscriber. Before the data is transmitted to cloud it is required that data is encrypted, there are different data security patterns in ZMQ as shown below:

(a) Grasslands, which is plain text with no security at all. All connections are accepted, there is no authentication, and no privacy

(b) Strawhouse, which is plain text with filtering on IP addresses.

(c) Woodhouse, which extends Strawhouse with a name and password check. For this, we use a mechanism called "PLAIN" (which does plain-text username and password authentication). It's still really not secure, and anyone sniffing the network (trivial with WiFi) can capture passwords and then login as if they were the real user

(d) Stonehouse, which switches to the "CURVE" security mechanism, giving us strong encryption on data, and (as far as we know) unbreakable authentication. Stonehouse is the minimum you would use over public networks, and assures clients that they are speaking to an authentic server, while allowing any client to connect.

(e) Ironhouse, which extends Stonehouse with client public key authentication. This is the strongest security model we have today, protecting against every attack we know about, except end-point attacks (where an attacker plants spyware on a machine to capture data before it's encrypted, or after it's decrypted).

In our architecture we will employ Stonehouse Curve 2 step encrypted security mechanism, 1st step of encryption whitelist the Ip address and the transmission takes place for only authentic IP addresss. Next public and private key certificates are being generated using generate certificates program at both Publisher and Subscriber, publisher should have the public key of subscriber to communicate through ZMQ and the subscriber sends the private key to the publisher and the connection is made. This is how 2 step encryption takes place.

C. Data Management

InfluxDB is a time-series database platform allowing to store the data in real time. It acts as a data store for large amount of time stamped IOT sensor data. We can query the data continuously and store it in InfluxDB in real time with flexible retention policies, so that we can choose for how much time we want to store a particular amount of data. It has a built in query platform InfluxQL similar to SQL. It allows us to filter out data within specific time window and perform some analytics including anomaly detection. Moreover, it is open Source and handles high data ingest and real time querying of sensor data. It also allows data compression on storage.

Grafana is an open tool platform for data visualization, analytics and data monitoring. It has inbuilt support for InfluxDB. It comprises of a dashboard with rich query support for plotting graphs in real time. So, through simple query the users can get a graphical look at the sensor data.

Thus we can store our sensor data in InfluxDB and plot it in Grafana, an instance of which is shown in detail in the subsequent discussions.

D. Use of computing paradigms

Various levels of computing resources have been used in our architecture, which includes cloud computing, community cloud and fog computing. Inspired by various works as described in the section 2, this work integrates these borrowed ideas. In this work some decisions have been made on optimally distributing the computation among the various available resources. As discussed the work revolves around using cloud, community cloud and fog. Our decisions about using the resources were according to the importance and the type of data being sensed in the city.

The data generated from the homes are big data considering the number of homes in the community and sensors in each home. This data also has some special requirements of security and privacy. Owing to the security requirements, we had to consider either using services of cloud vendors which are secure or either fog servers, which could be protected. On an average a 600 home community would sense about 200GB of data on a daily basis, so the capacity of the cloud vendors or dedicated servers could be used to store and process the data. Also, since this is not a safety critical application, we have no constraints on latency and this provides us a little flexibility on choosing cloud to be our primary computing resource.

So, by considering all the key requirements of privacy, security and latency, this work chooses to use cloud services of Amazon EC2 as the primary source of computing resource. From figure 2, it is seen that the data collected from the sensors are stored in a database running on an EC2 cloud instance.

However, as an extension to the primary cloud resource, we would also like to include fog servers as the secondary computing resource. So, the data generated at the sensors could be either processed at the fog server or at the cloud. We need to further investigate offloading techniques to distribute the data among fog and cloud.

However, looking at the nature of data collected from the air quality sensors, privacy, security and latency is not going to be vital and hence we can store it and perform analytics in the community cloud. As discussed before, the concept of community cloud works on trust, where resources of the community like phones, laptops could be used to form a p2p network and form a virtual data center to store data and perform anomaly detection.

The following sections discuss some of the results and architectural challenges of the novel architecture discussed above.

V. RESULTS

In the previous sections we have talked about the architecture and implementation of the smart home data streaming and analytics. This section would talk about our experimental setup and some results obtained.

A. Experimental Setup

Our setup to simulate smart home data streaming and analytic included:

(a) **Sensors:** Aeotec Multisensor6 which measures temperature, humidity, luminance, motion and ultraviolet values. This sensor works on the z-wave protocol and it interfaces with devices using the OpenZwave library.

(b) **Z-stick controller:** This is the controller which creates a Z-wave network and adds the sensors into it. Aeotec Z-stick was used as the controller in our implementation.

(c) **Home controller:** As discussed above Homeseer is an automation controller, which works as a data entry point. Since Homeseer is a linux box with Ubuntu 14.04 LTS, we have simulated it using a virtual box on a laptop running Ubuntu 16.04 LTS. Here every laptop (VM instance) acts as individual smart homes.

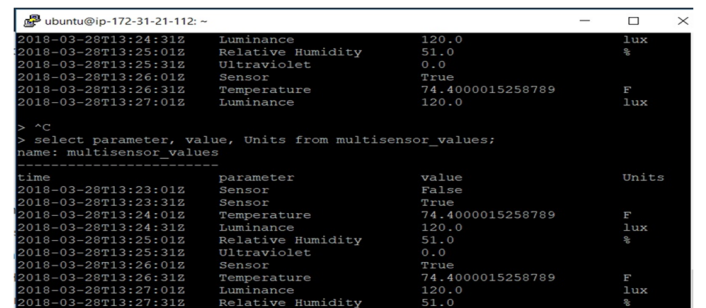
(d) **Data Streaming middleware:** ZMQ with ironhouse security is used for transporting the data from our laptop to the database. This demonstrates our requirements of having security while transporting the data outside the homes.

(e) **Data Management:** The data destination in our work includes several instances of influxdb which runs on the cloud. This also includes grafana which plots graphs of the data being registered in influxdb.

(f) **Computing resource:** As discussed in the previous section, a t2.micro linux instance from amazon EC2 is used as the data destination, which hosts influxdb and grafana.

B. Experimental results

Involving all the above discussed actors into the implementation we managed to obtain some initial results of data being sensed and then being stored in the influxdb instance. The figure below shows the data being registered in the influxdb instance which is running on Amazon EC2 instance.



```
ubuntu@ip-172-31-21-112: ~  
2018-03-28T13:24:31Z Luminance 120.0 lux  
2018-03-28T13:25:01Z Relative Humidity 51.0 %  
2018-03-28T13:25:31Z Ultraviolet 0.0  
2018-03-28T13:26:01Z Sensor True  
2018-03-28T13:26:31Z Temperature 74.4000015258789 F  
2018-03-28T13:27:01Z Luminance 120.0 lux  
> ^C  
> select parameter, value, Units from multisensor_values;  
name: multisensor_values  
-----  
time parameter value Units  
2018-03-28T13:23:01Z Sensor False  
2018-03-28T13:23:31Z Sensor True  
2018-03-28T13:24:01Z Temperature 74.4000015258789 F  
2018-03-28T13:24:31Z Luminance 120.0 lux  
2018-03-28T13:25:01Z Relative Humidity 51.0 %  
2018-03-28T13:25:31Z Ultraviolet 0.0  
2018-03-28T13:26:01Z Sensor True  
2018-03-28T13:26:31Z Temperature 74.4000015258789 F  
2018-03-28T13:27:01Z Luminance 120.0 lux  
2018-03-28T13:27:31Z Relative Humidity 51.0 %
```

Fig. 8. Influxdb data

Once the data is registered at the influxdb instance, it will be plotted with grafana, which is an open source plotting tool. The figure below shows different sensor data being plotted, the different data being plotted are temperature, relative humidity, burglar alarm and luminance.

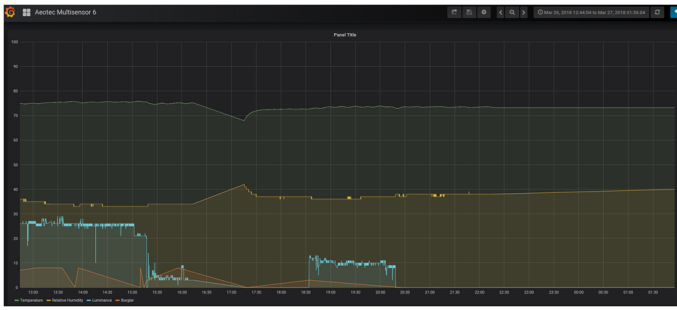


Fig. 9. Grafana plotter

VI. ARCHITECTURAL CHALLENGES

Analyzing our architecture, we have analyzed some of the important challenges which have to be emphasized while implementing our smart city architecture, they are:

(a) **Security:** With so much of data being generated and processed, maintaining privacy and data security throughout streaming and analytics is vital and challenging. The data should be encrypted/ anonymized to prevent attacks. We have some baseline security through ZMQ elliptic curve in data streaming phase and also ufw (firewall) protection at the cloud.

(b) **Reliability:** Its a big challenge to make our architecture reliable, and to make sure none of our actors fail. To encounter this at the cloud we have elastic ip (floating ip) which on the fly attaches to a new instance in case of failure. Also at the streaming end, ZeroMQ ensures successful message transmission/receipt over TCP. To provide an overall reliability architecture, we plan to include Resilient Information Architecture Platform for the Smart Grid (RIAPS) over which we could run our smart home application. RIAPS is a promising platform which includes some important features like fault management, persistence, resource management and discovery.

(c) **Plug & play:** Incorporating heterogeneous sensors into the framework is challenging. As of now we have support for Z-wave protocol, but we need to extend support to devices which work over bluetooth and WiFi. Also the plug & play feature should work without making any changes into the architecture.

(d) **Location Transparency:** Irrespective of the location of the data storage, the data storage and analytic server should be available anywhere and always to the users. As discussed before, since we have multiple computing resources involved, our servers have to be available at different places without any problem.

(e) **Data Offloading:** Data offloading has been talked for many years now. There have been different literatures performing offloading based on different QOS requirements. Similarly we need to figure out a robust offloading technique

to distribute the data between cloud and the fog servers to reduce latency, cost and delays.

(f) **Feasibility of Community Cloud:** Since the concept of community cloud is still naive, the question arises about the feasibility of community cloud and applying it into our architecture. As discussed in section 2, there is still many critical questions yet to be answered before using the services of community cloud.

These are some of the challenges which was seen while working with the implementation, some of them have already been taken care of while some others have to still be taken care off in the future.

CONCLUSION & FUTURE WORK

This work aims at providing insights into building data streaming and management architecture for smart cities. An in depth analysis has been done in areas of streaming, data management, analytics and computing paradigms to build the novel architecture discussed above. The critical selection of the actors involved in the architecture has been done based on the application requirements. A naive implementation of smart home data streaming and management has also been discussed above along with some results.

As an extension to this work, we need to put in some critical thinking into performing computation offloading, instance migration between computing paradigms, and distributed data analytics. Along with this we need to address the architectural challenges discussed above.

REFERENCES

- [1] Zwave Alliance. Python-openzwave v0.4. Last accessed April. 2018.
- [2] K. Benson, C. Fracchia, G. Wang, Q. Zhu, S. Almomen, J. Cohn, L. Darcy, D. Hoffman, M. Makai, J. Stamatakis, and N. Venkatasubramanian. Scale: Safe community awareness and alerting leveraging the internet of things. *IEEE Communications Magazine*, 53(12):27–34, Dec 2015.
- [3] G. Briscoe and A. Marinos. Digital ecosystems in the clouds: towards community cloud computing. In *IEEE 2009 the 3rd IEEE International Conference on Digital Ecosystems and Technologies (DEST 2009)*, pages 103–108, New York, USA, 2009.
- [4] Marisol García-Valls, Tommaso Cucinotta, and Chenyang Lu. Challenges in real-time virtualization and predictable cloud computing. *Journal of Systems Architecture*, 60(9):726 – 740, 2014.
- [5] Peter Mell and Tim Grance. The NIST definition of cloud computing, v15, NIST. 2009.
- [6] M. R. Rahimi, N. Venkatasubramanian, S. Mehrotra, and A. V. Vasilakos. Mapcloud: Mobile applications on an elastic and scalable 2-tier cloud architecture. In *IEEE/ACM 5th International Conference on Utility and Cloud Computing (UCC12)*, pages 83–90, Chicago, Illinois, USA, Nov 2012.
- [7] Mahadev Satyanarayanan, Pieter Simoens, Yu Xiao, Padmanabhan Pillai, Zhuo Chen, Kiryong Ha, Wenlu Hu, and Brandon Amos. Edge analytics in the internet of things. *IEEE Pervasive Computing*, 14, April 2015.
- [8] J. M. Schleicher, M. Vogler, C. Inzinger, and S. Dustdar. Towards the internet of cities: A research roadmap for next-generation smart cities. UCUI@CIKM, 2015.
- [9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. In *IEEE Internet of Things Journal*, volume 3, pages 637–646, October 2016.

- [10] M. Y. S. Uddin, A. Nelson, K. Benson, G. Wang, Q. Zhu, Q. Han, N. Alhassoun, P. Chakravarthi, J. Stamatakis, D. Hoffman, L. Darcy, and N. Venkatasubramanian. The scale2 multi-network architecture for iot-based resilient communities. In *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8, May 2016.
- [11] S. Yi, C. Li, and Q. Li. A survey of fog computing: Concepts, applications and issues. In *Proceedings of the 2015 Workshop on Mobile Big Data. ACM*, 2015.