Architectures and Data Processing Algorithms for IOT Gateways

Fu-Hsuan Liu
Electronical Engineering
National Taiwan University
Taipei, Taiwan
b03901191@ntu.edu.tw

Sheng-De Wang
Electronical Engineering
National Taiwan University
Taipei, Taiwan
sdwang@ntu.edu.tw

ABSTRACT

The Internet of Things (IOT) technology is gaining increasing public attention, and IOT devices will be more ubiquitous in the future. Coming along with the growing number of data sources, we need to face some challenges such as heterogeneity of data types and protocol incompatibilities. Moreover, to minimize the communication time and enhance the performance, sending entire large amount of data is not practical. These factors give rise to edge-computing, or data pre-processing on IOT gateways. In this paper, we propose a data processing architecture and a data reduction algorithm for the gateway of bridge vibration G-sensors. The architecture also considers the features such as the system security authorization and real-time data visualization.

Keywords: Internet of things, gateway, MQTT, edge computing

1. INTRODUCTION

Due to the booming development of IoT, various sensor devices, modules and embedded systems are connected at the same time in order to reach its multifunctional feature. The issue of a huge amount of data with heterogeneous types becomes a challenge. The infrastructure of conventional IOT systems is lack of a mediator between devices and clouds. The concept of edge computing such as data pre-processing on a gateway device is needed. Data reduction in a gateway can prevent I/O bottlenecks and significantly reduce cloud storage consumption. It can also lead to a reduction of transmission time, network bandwidth requirements and energy costs. An intelligent gateway whose role interconnected end devices to central of system infrastructures can fulfill the goals of device management and data storage reduction by edge computing [1].

There are many existing gateway structures designed for a variety of use. It is important to choose appropriate data collection protocols for the IOT gateway. Masek et. al. [2] introduces a wide range of networking protocols used in IoT. Every protocol has its own feature. For example, SIP, one of the main VoIP protocols, is in applications of internet telephony for voice and video calls; CoAP, another service layer protocol used in wireless sensor network nodes, has strong points in working with HTTP in the web; LoRa is a wireless communication solution which is superior in its long-distance and low-rate features. A LoRa gateway and LoRa network server is introduced in [3] to pre-process the data and overcome the problem of high volume problem; MQTT is the most lightweight TCP/IP IoT protocol designed for connections with remote locations where network bandwidth is limited, also known for its low frequency and computing demand. Therefore, MQTT is widely used in IoT gateway. For example, an IoT home Gateway based on MQTT can automatically configure and manage the devices under a home energy-saving scenario [4].

Another crucial reason for implementation MQTT to a gateway system is its "publish-subscribe" method, which enables a large flexibility both for sensors and users. In the publish-subscribe pattern, a message broker acts as a middleman in need for distributing messages between publishers and subscribers based on 'topic', as shown in Figure 1. A publisher can publish a message to a broker, and only subscribers who know the corresponding topic can subscribe, and the QoS of MQTT offers user three kinds of data transfer quality that can be defined based on different applications.

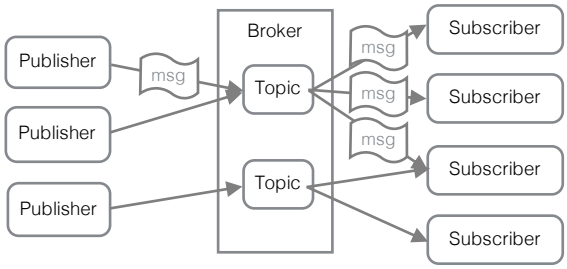

Figure 1. The concept of MQTT.

This kind of "twitter-like" communication protocol pattern can increase the number of features (or called function controllers) in the gateway system. By making the gateway an MQTT broker, we can attach sensor devices to the gateway and data consumers can subscribe the data provided by the sensor devices. We can also add functions we need as a subscriber to the broker. Thus, the gateway can serve as a many-to-many communication platform, which can enable M2M message exchange to receive data while subscribed by different subscribers with different function. The design meets our demand for the gateway architecture – intelligent and multi-functional.

In this paper, we propose an MQTT-based intelligent gateway, where many "functions or data processors" can be put forward for the purpose of the data processing. As an example, we particularly propose an application for bridge vibration detection, where a data processor implements a data-reduction algorithm aimed to processing vibration data. An IOT system always need to process huge amount of data to meet the rigorous analysis need and to store these data in real-time. The data processor can extract the data patterns we need in the data streams and thus the data storage can be reduced to the amount to keep only the data patterns. Therefore, the reduction in the data quantity can markedly cut the storage cost.

The application of the bridge vibration detection needs to process the data in real-time and the data are coming in a data stream or a time series style. Lee et. al. [5] propose an algorithm for reducing the size of time series data based on the RETE algorithm for many

motes sensors. The main concept is event-basis and all data have been processed before reach to the gateway. The gateway cannot do data aggregation at the same time. For example, sometime we want to send event-basis data to the storage, while visualize the data in real-time.

In addition to the goal of reducing the size of data storage or the time cost of transmitting, we sometimes require the data synchronization. Timestamp and bitmap are introduced to reduce the length of data and to synchronize in an IOT gateway [6]. By implementing timestamp and bitmap, it can just update the current data rather than re-update the whole data set. However, the problem solved by such algorithms is trivial, since the cache size is always proportional to data size.

Papageorgiou et. al. [7] propose an edge-computing method that focuses on the IOT data analysis and on handling time series data without adding significantly delay. The method makes use of an agent combined with multiple functional data processing handlers. The reduction of real-time data size is accomplished by a Perceptually Important Point (PIP) - based algorithm. The concept can be applied to data streams can reduce the cache size while not affect the accuracy of data analysis.

In this paper, we not only introduce an intelligent gateway architecture but also propose a data processor for the application on the bridge vibration detection. The data processor demonstrates one of gateway features, which is to categorize the data into patterns, and send the data patterns to the cloud storage. The results show the effectiveness of our approach.

2. APPROACH

2.1 Gateway System Architecture

An intelligent gateway based on the MQTT protocol can form a multi-functional system by its flexible "subscribe - publish" pattern, as shown in Figure 2. The multi-functional system is composed of the MQTT broker and various functional processors or controllers. Despite the devices connected to the gateway by the ways of either Wi-Fi, LoRa, BLE or wired connection, the data sources or the data consumers can be seamlessly integrated. Once a device connects to the gateway, it is managed by the controllers and the generated data can be processed by the data processors or the controllers.

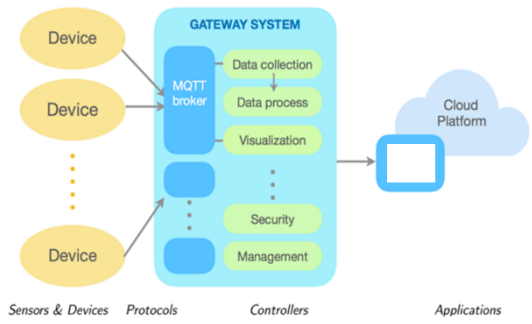

Figure 2 Gateway system architecture.

A list of multiple data-processing-features is shown in Table 1. These function controllers include: Data collection, Cache storage, Event detector, Real-time visualization, Security authorization and Control and management. With these functions, we can manage devices with various data types under a certain security degree. Furthermore, we only need to modify the basic settings on the gateway, for instance the data sampling rates and condition of sensor nodes, but do not need to modify for each device, which really saves time in the device management. We also demonstrate data processing algorithms that can reduce the amount of data, while maintaining the data semantics that keeps the quality of data and accuracy of results.

TABLE 1 GATEWAY CONTROLLERS FEATURES.

Gateway processor system	
controller	**function**
Data collection	Can collect data value form devices, sensors and modules by using MQTT, API protocol, or just by cable line. And the data will be collected and transmitted through a data-reduction algorithm while do not affect accuracy of results.
Cache storage	Record the new-in data value also its timestamp. A cache will be used on event-data set.
Event detector	When value of incoming data **meets the conditions** (e.g. value > threshold), we **called an "event" occur**.
Real-time visualization	Real-time visualize the streaming data (on browser) by build a WebSocket server on the gateway while client on a cloud platform (e.g. GCP).
Security & authorization	Based on protocol MQTT on data collection and transformation to platform, only devices and platforms who have 'topic' (which act like a key) can access the gateway and participate in these processes.
Control & management	Including two main features: 1. Sampling processor - To determine the data sampling rate for each data source. 2. Sensor node condition setting – To decide each of sensor nodes is offline or online hence reduce the power consumption.

2.2 A Controller Application: Streaming Data Reduction Algorithm

Sensor devices usually run all day long in order to monitor in real-time and track its daily condition every time, every moment. However, except for some event cases occur occasionally, data collected over a long time in general has little change. If we can assure that the data reduction result will meet the goal, recording the whole data set all day long is meaningless. We need only to track its trend or condition. To monitor and keep the condition of bridges, we design a data classification method to extract the vibration patterns from the data streams. The data classification method can target at the detection of large vibration such as earthquake events.

The data reduction algorithm is based on the classification of vibration data and the detection of some special events. One of goals of edge-computing is to process data, pre-analyze, and reduce the number of data we need to upload. There are mainly two parts in our data reduction process, the *event-basis case* and the *general interval case*. Due to the feature of bridge vibration, categorizing the events into two cases can improve the performance and satisfy our need. In the event-basis case, which means an event happened in the duration, we need to record the whole set of data since we do not want to miss any detail. However, in the general interval case, we should present analysis results as simple as possible to reduce data size.

1) *Event-basis case*

An event means that the incoming data meets some conditions. In our project, we focus on streams of vibration data and an event is defined as when incoming data's values are more than a threshold value during an interested duration.

When an event (like an earthquake) occurred, the data before and after the event will be kept for record, that is will be uploaded to a cloud storage. We need to preset a duration to hold the data. To this end, we use a first-in-first-out queue to keep the data streams, as shown in Figure. 3. If the data associated with the detected event travels to the middle of the data queue, we then record the whole data queue. Because this is event-case, the data should be kept it real and complete.

The setting of the threshold value and the duration are based on the empirical field knowledge.

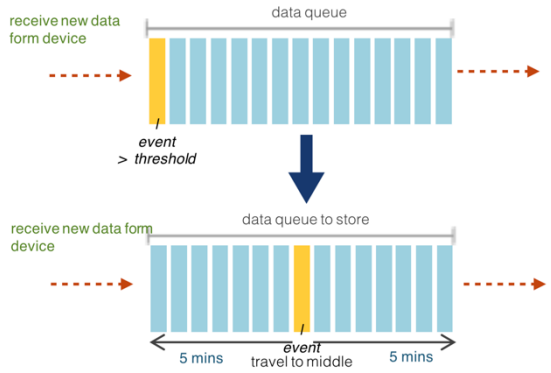

Figure 3 Event-basis data reduction algorithm

Pseudo code for Pattern System- event-basis case

```
Input: new_in_data
Output: event_set
-----------------------------------------------------
DEFINE cache_length = 10 mins * sampling rate;
DEFINE threshold_value = 0.2Gal;

While new_in_data travel through cache // FIFO
If new_in_data.value > threshold_value
    When new_in_data traveln to middle of
    cache_length
        event_set = cache; // cache right now
store event_set
```

2) *General Interval case*

In this case, we design a data classification algorithm to categorize the data streams into patterns. In the algorithm, we learn from existing datasets for a set of patterns, and each pattern represents the whole data set of a time interval, which is user-specified on a gateway. The dataset for the learning of patterns is collected before the classification for a period of time, denoted as P minutes. The dataset to represent current data streams is collected for an interval of time, denoted as Q minutes. The general interval case means the income vibration data may be similar for a period time and can be represented by existing patterns.

In this project, an input data type of a time-series-data is a 3-tuple, (x, y, z), where (x, y, z) stands for the acceleration value of each axis. During the **first step**, we take its vector value of each axis as the input to the algorithm. The values of P and Q are set by field knowledge and experience and may be different in various fields or based on user customization. The concept is shown in Figure 4. This P-mins dataset is transformed into a 3*M matrix, where 3 means the three-axis, while M means that the number of data during the P-minutes is M. Therefore, a point on behalf of the P-min-dataset positioned in the space.

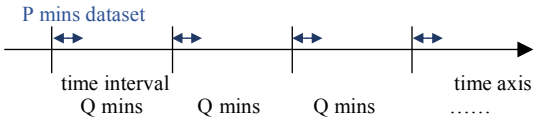

Figure 4 General interval case time section diagram.

In the **second step**, we use **K-means algorithm** to partition the dataset into k clusters. The K-means algorithm is an iterative refinement technique suitable for unlabeled sensor data [8]. It's an algorithm to find K clusters which have the shortest distance to the data points in the dataset. And data points are categorized into these clusters. Here we briefly introduce the concept of K-means and motivation of why it is adopted:

1. The K-means algorithm is fast
2. The K-means algorithm has good scalability to even large dataset, which is often used to find a configuration.
3. clustering analysis: categorize unlabeled input data into clusters based on feature similarity.

The core concept is to find the minimum sum of points in S_i to the set of K cluster centers:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \qquad (1)$$

where μ_i is the mean of data items in S_i.

Since it applies to unlabeled data, the K-means algorithm is an unsupervised learning. There are other reasons why we select this algorithm.

A P-min dataset will divide into R sections. In our test system, P, Q, R equal to 3 minutes, 30 minutes, 3 respectively. In applying the K-means algorithm to learn the patterns, each resultant cluster represents a vibration pattern. We can determine the value of initial k by empirical, which is field-dependent, or by **elbow method** [9].

However, it's a challenge that since streaming data will always present some form of concept drifts, which are defined to be the distribution of data changes over time [10]. That is, in most of the time without concept drifts, new coming time series data remains unchanged or similar to previous dataset, dataset is categorized to one of the existed pattern clusters; if there occur concept drifts, the current data point would be far from the previous pattern clusters, a new pattern set needs to be defined and built. As a result, the pattern set and value of k should be updated once the concept drift is detected.

Therefore, in the **third step**, in addition to the K-means algorithm for clustering, we update the number of clusters, or k, and the model, which is the pattern set,

by using a **concept drift detection** method. The concept drift detection serves as an enhancement for the streaming data classification. A drift is detected if the average of the sum of distances between clustered data points and the center of the belonged cluster is bigger than a threshold value. Then the k in K-means algorithm and the pattern set need to be updated. And the gateway will also record the time when the drift is detected. The following coming dataset will use this new k in K-means for processing.

After clustering by K-means, the cluster labels are given to represent the data streams. In the vibration data classification, the pattern will be represented in 3 sections respectively for 1min-1min-1min, for example, a-c-d. For a Q-minutes interval, the data streams will be represented by a series of data patterns, which is learned from the P-minute dataset. The whole process of data reduction is shown in Figure 5.

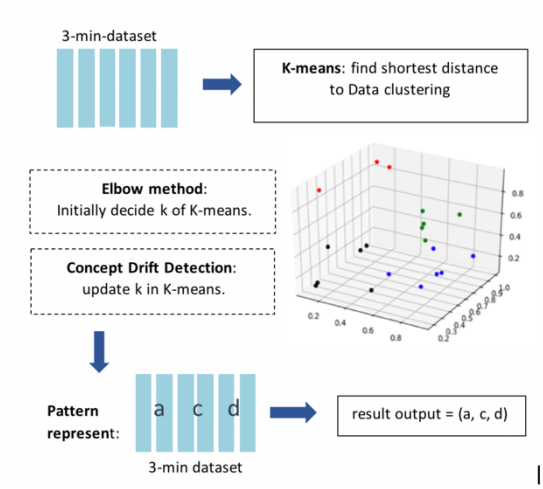

Figure 5 illustration of Pattern System.

Pseudo code for Pattern System- Concept drift detection

```
Input: new_in_ds // split into 3 sections
-------------------------------------------------
DEFINE threshold_sumOfSquares = v
DEFINE threshold_lastTime = s

K-means(n_cluster = k).fit(new_in_ds)
For each cluster in Clusters:
   For each point in cluster:
// Sum of squares between each point in cluster
    and cluster center
      sqSum += square distance between (point, cen-
                          ter)
   avgSum(cluster) = sqSum / (# of points)
Sum = sum(avgSum(cluster))
If Sum > v:
   time_counter = time_counter + 1
    //Sum > v and last for a certain threshold of
time
    If time_counter = s:
      k = k+1 //update k
      time_counter = 0
      update pattern-library
      report update k, timestamp
```

3. IMPLEMENTATION

In this project, we design a scenario to test the system. Figure 6 shows four ways of sending data from three sensor nodes to the cloud platform: MQTT (wifi) via gateway, LoRa via gateway, cable line via gateway and not via gateway. We take ADXL345 and LinkIt Smart 7688 Duo as a sensor node prototype, and a sensor node consists of three-axis accelerator prepared by CIC (National Chip Implementation Center). A Raspberry-Pi 3 (R-pi 3) acts as a gateway to receive data from devices. We compare the results of data collecting system with gateway and without gateway. Furthermore, we also show the extension capability of the proposed gateway by adding the functionalities of the realization of visualization via WebSocket. With the gateway, we can easily manage many devices, process data, maintain security and fulfill visualization, most important, reduce data number and still maintain accuracy.

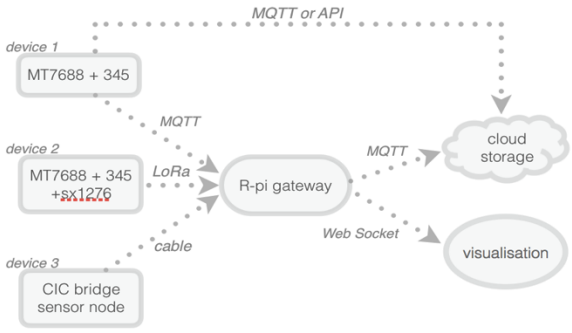

Figure 6 Framework of project scenario.

3.1 MQTT Data Collection

To demonstrate the practice of the proposed gateway, we implement the gateway using a lightweight single board computer, RPi 3. The implemented "subscribe-publish" pattern to transfer data with the MQTT protocol is shown in Figure 7.

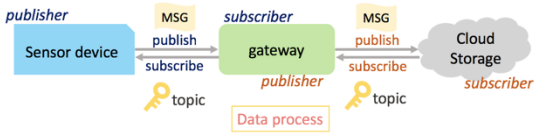

Figure 7 flowchart of MQTT data collection.

A message sent from device end through the gateway to cloud side can be illustrated in two parts: *device to gateway* and *gateway to cloud*.

1) Device – Gateway

During this data transmitting process, Device 1's MPU is an MQTT publisher, the gateway acts as an MQTT broker and a subscriber. An MQTT broker 'Mosquito' provides publish-subscribe function via the 'Topic' to send/get message from devices. The topic acts like a key, one who knows the key can subscribe or publish.

2) Gateway - Cloud storage

Here gateway acts as data provider, and a cloud side acts as a data subscriber. Again, the process is similar to *Device - Gateway*, through an MQTT broker. The messages sent to the cloud are data after filtered by the gateway, which we have discussed it in the previous section.

3.2 Data Reduction Algorithm

We design the algorithm based on the specialty of time series data (x, y, z), after a 3-min dataset collected in the cache, it will be split into 3 sections, which is 1min-1min-1min in our scenario. A row of data ([x, y, z]) represents a point. we approach K-means algorithm to process data in order to define the "pattern" of a dataset of static time section, and a concept drift detection is designed to update the pattern library and diminish the error clustering result from concept drift. Finally, we only need to present and store "pattern" rather than store whole data set to server.

Furthermore, an event-detector does not conflict with the general-interval case, when knowing an earthquake may occur, we then record how it last. A threshold value is specialized as 0.2 Gal (= 1cm/s^2 = 0.098 g). Once we detect the value of a data > Threshold, we record the data 5 minutes before and after that event, totally 10 minutes. Since this is the event-case, we want to store the most comprehensive and the complete information, sending the raw data immediately to cloud platform for visualization is needed.

After data pre-processing at edge-computing, we even make the results more descriptive.

3.3 WebSocket Visualization

Real-time visualization of the result is one of our gateway system feature. We implement WebSocket to send data, and "Smoothie Charts", a library which enables us to establish a real-time waveform on web, unlike other real-time visualization modules, it doesn't squeeze the waveform with the increase of input data. Rather than a condemn waveform which has a smaller scale of data and a significant delay for re-scaling the whole figure when updating every time, it's a moving window in real-time, the scale remains unchanged and in real-time.

We create a WebSocket server on a gateway, and client on a webpage, as shown in Figure 8. Once a connection is built between a server and a client, an HTTP Request is sent from the client to the server; after server update WebSocket protocol, Handshake is established to enable a bi-directional message transmission. In the html file, assigning server IP is required to connect to the WebSocket server. The waveform results will show on a browser page. A sample result will be shown in section V.

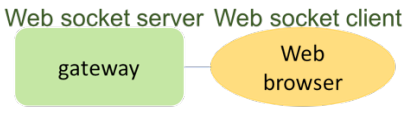

Figure 8 Implementation of visualization controller.

4. EXPERIMENTAL RESULTS

4.1 MQTT data collecting

Here we compare the performance of sending data under different protocols in order to observe distinct protocols integrated with the proposed gateway. And the capabilities of several cloud storage platforms under our application scenario are also shown in Table 2.

TABLE 2 TRANSMITTING TIME WITH DIFFERENT PROTOCOL UNDER THE GATEWAY SYSTEM

Device	Transmit method	Time(s)
MT7688 + ADXL345	LoRa	2.85
	MQTT(wifi)	1.62
	API	2.09

TABLE 3 PERFORMANCE OF DATA TRANSMISSION

Transmit-ter	Receiver	Transmit Method	Transmis-sion Rate	Packet Loss Rate
Device	Gateway	MQTT	1.6s/a data	0%
		LoRa	2.9s/a data	4%
	Data Store (MCS)	MQTT	2s/a data	0%
		API	3s/a data	6%
	Data Store (Thing-Speak)	MQTT	2s/a data	84%
		API	3s/a data	84%
Gateway	VM	MQTT	2s/a data	0%
VM	Data Store (MCS)	MQTT	2s/a data	0%
		API	3s/a data	0%
	Data Store (Thing-Speak)	MQTT	2s/a data	86%
		API	3s/a data	84%

We can see that MQTT (wifi) has better performance at most time, such as the data transmitting time. But 'ThingSpeak' only designs to support very limited number of protocols and will have a lot of packet loss and a serious delay.

4.2 Data reduction algorithm

A piece of screen shots is demonstrated in Figure 9. There is 3000 input streaming data with the datatype (x, y, z), which form a matrix with size equals to [3000 * 3]. After K-means and the concept drift detection, a pattern-series result is yielded as (A, C, D).

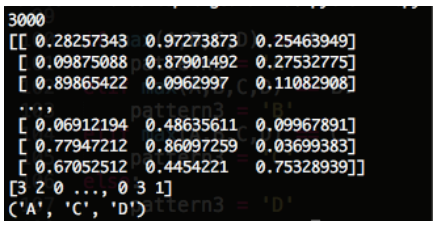

Figure 9 Screen shot of a data reduction result.

We compare the performance of data-reduction controller to the original raw data. Assume that if we record a 3-minutes queue data every 30 minutes, and the sampling rate equals to 200Hz (SR). In summarize, the size of the queuing data we want to send to a server per package are:

Case 1- normal case (for one hour):

*3600 secs * 0.1 * SR * 3 bytes per pattern sequence = 1.08 KB*

Case 2- when an earthquake occurs (for one time):

*10 mins * 60 secs * SR * 2KB per data = 2.4 MB*

It is estimated that a usual day without powerful vibration, which is a normal case, needs about *1.08 KB*, but if we record all the time, it needs (for one hour):

*3600 sec * SR * 2KB = 3456 MB = 1.44 GB.*

From the result we can see that the memory usage approximately 1333 times lesser than the original raw data requirement.

4.3 WebSocket Visualization

Multiple devices can connect to our intelligent gateway with different protocols at the same time. We design a visualization controller to show the data values as a use scenario: Device1 (LinkIt Smart 7688 Duo with sensor ADXL345) connect via wifi (MQTT) and Device2 (LinkIt Smart 7688 Duo with sensor ADXL345 and sx1276) by LoRa connection. The sensor devices are setup in different locations with distinct conditions, and both connect to Raspberry Pi 3 gateway. Assume that the sensor nodes are both given a shock move and reach a threshold value. They are categorized into event-case and shown in the browser. For a simplified result, we turn (x, y, z) three input parameters into one resultant value as output values. In Figure 10, x-axis labels as time and waveforms different in colors distinguish different devices.

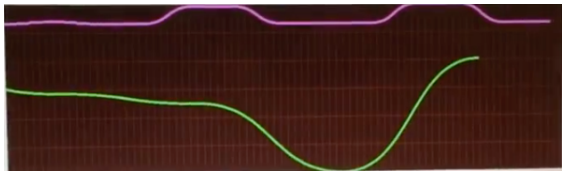

Figure 10 Visualization on a web browser via smoothie chart: data of two sensor devices at different conditions.

5. CONCLUSIONS

The proposed MQTT-based gateway architecture enables us to build a flexible multi-functions system. Time-series input data is processed by K-means algorithm followed by concept-drift detection. As a result, we get a set of "patterns" to describe the whole dataset during the time duration. This kind of data reduction is appropriate to a class of real-time input data if we care about the patterns in the data belongs to. In this case, we can keep the patterns at the cloud and thus reduce the cloud storage requirement successfully.

REFERENCES

[1] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, Michele Zorzi "Internet of Things for Smart Cities", n: IEEE Internet of Things Journal (Volume: 1, Issue: 1, Feb. 2014)

[2] Pavel Masek, Jiri Hosek, Krystof Zeman, Martin Stusek, Dominik Kovac, Petr Cika, Jan Masek, Sergey Andreev, Franz Kröpfl "Implementation of True IoT vision: Survey on Enabling protocols and Hands-on Experience", International Journal of Distributed Sensor Networks, 2016.

[3] Jaeyoung So, Daehwan Kim, Hongseok Kim, Hyunseok Lee, Suwon Park, "LoRaCloud: LoRa Platform on OpenStack", NetSoft Conference and Workshops (NetSoft), 2016 IEEE

[4] Seong-Min Kim, Hoan-Suk Choi, Woo-Seop Rhee, "IoT home gateway for auto-configuration and management of MQTT devices" , Wireless Sensors (ICWiSe), 2015 IEEE Conference on

[5] Changmin Lee, ChangHyun Byun, Hyungjeong Shin, "A resource-efficient algorithm for processing various sensor data in Internet of Things applications", Advanced Science and Technology Letters Vol.136 (ITCS 2016), pp.100-102

[6] Zhijie Lin, Lei Zhang, "Data synchronizing algorithm for IOT gateway and platform", 2nd IEEE International Conference on computer and communications, 2016.

[7] Apostolos Papageorgiou, Bin Cheng, Erno Kovacs, "Real time data reduction at the network edge of internet-of things systems". 978-3-901882-77-7, 2015 IFIP

[8] H. Hromic, D. L. Phuoc, M. Serrano, A. Antonić, Ivana P. Žarko, Conor Hayes, Stefan Decker "Real time analysis of sensor data for the Internet of Things by means of clustering and event processing" Communications (ICC), 2015 IEEE International Conference on

[9] Trupti M. Kodinariya, Dr. Prashant R. Makwana "Review on determining number of Cluster in K-Means Clustering" International Journal of Advance Research in Computer Science and Management Studies, Volume 1, Issue 6, November 2013

[10] Heng Wang, Zubin Abraham "Concept Drift for Streaming Data". International Joint Conference of Neural Networks 2015

[11] Smoothie Charts – A JavaScript Charting Library for streaming data
http://smoothiecharts.org/

[12] MediaTek LinkIt Smart 7688 Duo documentation
https://docs.labs.mediatek.com/resource/linkit-smart7688/en/documentation

[13] MediaTek Cloud Sandbox
https://mcs.mediatek.com/zh-TW/

[14] ThingSpeak
https://thingspeak.com/