Architectures and Data Processing Algorithm for IOT Gateways

Fu-Husan Liu, Sheng-De Wang

Electrical Engineering
National Taiwan University

b03901191@ntu.edu.tw

December 15, 2017

Overview

1. Introduction
2. Approach
   - Gateway Architecture
   - Algorithm of Streaming Data Reduction
3. Implementation
   - System framework scenario
   - MQTT Data Collection
   - Data Reduction Algorithm
   - WebSocket Visualization
4. Experimental Results

Introduction

- IoT devices will be more ubiquitous; growing number of data sources
  $\Rightarrow$ heterogeneity of data types and protocol incompatibilities

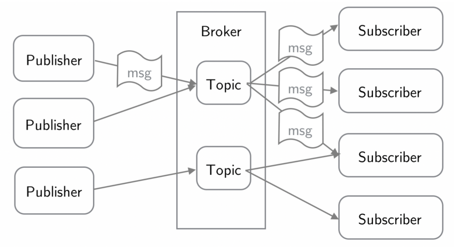
- minimize the communication time and enhance the performance
  $\Rightarrow$ sending entire large amount of data is not practical.
  $\Rightarrow$ give rise to edge-computing, or data pre-processing on IoT gateways.

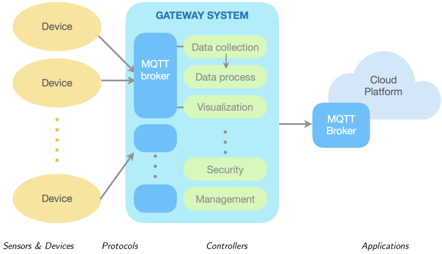
- In this paper, we propose a data processing **architecture** and a **data reduction** algorithm for the gateway of bridge vibration G-sensors.

- The architecture also considers the features such as the **system security authorization** and **real-time data visualization**.

Gateway Architecture

- **MQTT-based protocol**: publish-subscribe, flexible, leads to a *multi-functional* gateway structure.
- **Integrate different protocols**: WIFI, LoRa, BLE,... included.





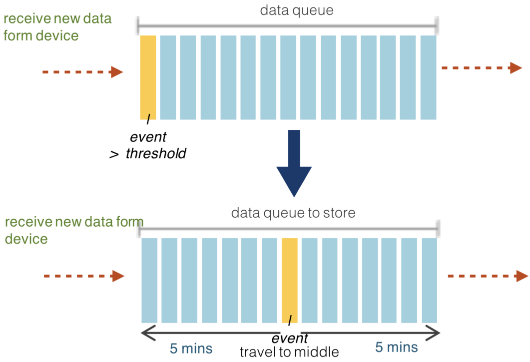
Algorithm of Streaming Data Reduction: Overview

The data of bridge vibration G-sensors we collect can be classified into 2 cases to process:

- **Event-base case**: When incoming data meets some conditions (threshold value)
- **General Interval case**: Normal, most of the cases, data always remain similar or unchanged.

Event-base case

- **Concept**: want to know the complete data set when an event occurred, in this project we record the data set for 5 minutes before and after of the event occurs.



Event-base case (cont.)

- Input: new in data (streaming); Output: event set.

Pseudo code for Event case

While **new_in_data** travels through cache:

  If new_in_data.value $>$ threshold:

    when new_in_ data travels to **middle** of cache length

      event set $=$ whole cache at that time

store event set and timestamp.

General-interval case

1. A "**Pattern System**" is designed as a data classification algorithm in this part.
2. 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.
3. Input streaming data type: **(x, y, z)**(g)
4. Four steps:

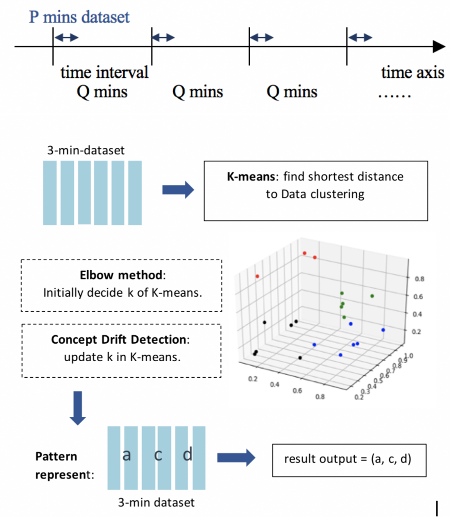
- **Step 1**. P, Q time interval dataset & 3×M-Matrix :[(x,y,z)]
- **Step 2**. K-means algorithm
- **Step 3**. Concept drift detection
- **Step 4**. Output Pattern-series result

General-interval case (cont.)

- **Step 1** P,Q time interval
- (optional) elbow method find k
- **Step 2** K-means

$$argmin_S \sum_{i=1}^{k} \sum_{x \in S_i} || \, x - \mu_i \, ||^2$$

- **Step 3** Concept drift detection: distribution of data changes over time, k need to be updated for enhancing the classification result.
- **Step 4** Series output

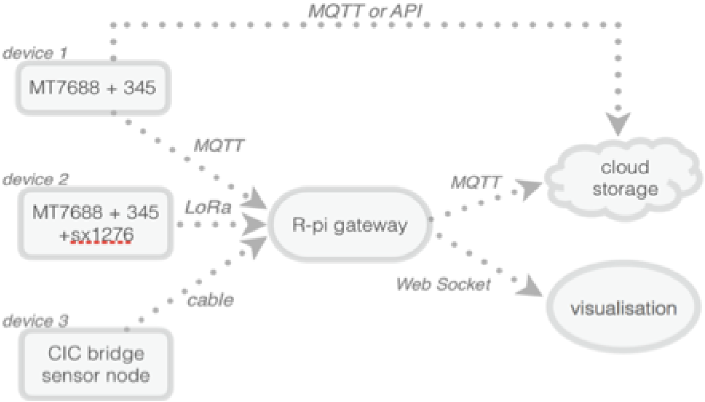


General-interval case (cont.)

Pseudo code for **concept drift detection** in general case

K-means(n_cluster = k).fit(new_in_ds)
for each **cluster** in Clusters:
    for each point in **cluster**:
        sqSum += square distance between (point, center)
    avgSum(cluster) = sqSum ÷ (# of points)
Sum = sum(avgSum(cluster))
If **Sum** > **v**:
time_counter = time_counter + 1
    If **time_counter = s**:
        k = k+1
        time_counter = 0
        update pattern-library
        report update k, timestamp

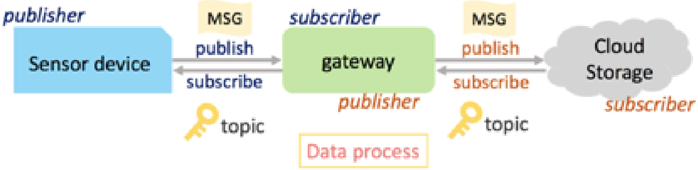
Implementation - Framework of project scenario



Implementation - MQTT data collection

There are 2 parts:

1. device - gateway
   An MQTT broker 'Mosquito' provides publish-subscribe function via the 'Topic' to send/get message from devices.
2. gateway - cloud storage



General interval case:

We design the algorithm based on the specialty of time series data (x, y, z):

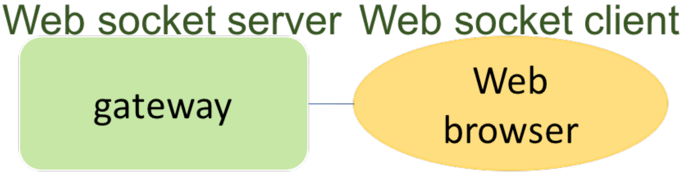
1. A row of data ([x, y, z]) represents a point.
2. after a 3-min dataset collected in the cache, it will be split into 3 sections, which is 1min-1min-1min in our scenario.

Event case:

1. An event-detector does not conflict with the general-interval case.
2. A threshold value is specialized as 0.2 Gal $(= 1cm/s2 = 0.098\ g)$.
3. Once we detect the value of a data $>$ Threshold, we record the data 5 minutes before and after that event, totally 10 minutes.

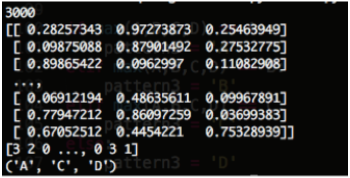
WebSocket Visualization

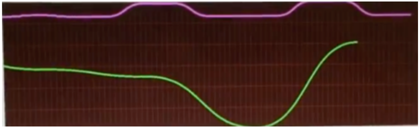
- one of our gateway system feature
- We create a WebSocket server on a gateway, and client on a webpage.
- Once a connection is built between a server and a client, an HTTP Request is sent from the client to the server;
- after server update WebSocket protocol, **Handshake** is established to enable a bi-directional message transmission



Experimental Results

1. Data Reduction
   - **Normal case** (1hr):
     *3600 secs * 0.1 * SR * 3 bytes per pattern sequence = 1.08 KB*
   - **Event case** (1 time):
     *10 mins * 60 secs * SR * 2KB per data = 2.4 MB*
   - **No algorithm** (1hr):
     *3600 sec * SR * 2KB = 3456 MB = 1.44 GB.*
     memory usage approximately 1333 times lesser!
2. Visualization:





Conclusion

- 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.

The End