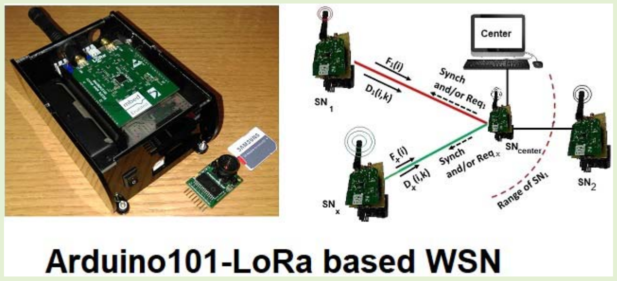
LoRa-Based Anomaly Detection Platform: Center and Sensor-Side

Mehrdad Babazadeh

***Abstract*—The implementation of a LoRa-based wireless sensor network, aiming to operate as an edge-based anomaly detection system is elaborated in this paper. The dominant prerequisites are defined as achieving a maximum possible sensing rate on edge, minimum Communication packet loss, the lowest possible false alarm, and the optimal resource management. In this regard, the developed sensor-side algorithm manages measurements accordingly to make effective communications with the center while transmitting either notification or requested compressed data packets. On the other side, a central algorithm keeps all the authorized sensors synchronized and updated. Besides, the center avoids network congestion primarily by queuing the received event notifications and later, by transmitting the data queries consecutively under supervision. The re-construction of the received anomaly-related data packets for further evaluations is another task to do by the center. Overall, the complementary proposal is experimentally tested and validated through the network operation in a worst-case scenario.**



***Index Terms*—LoRa, anomaly detection, edge analytics, wireless sensor network.**

I. Introduction

THE wireless sensor networks (WSNs) have been rapidly developing during the last decade. Yet researchers try to support this platform as a preferred choice in many applications such as the monitoring and anomaly detection systems because of the flexibility, low costs, and being retrofitted in difficult to access areas. They have been deployed through the limited-size projects and their applications have ramped up toward replacing the traditional wired platforms like supervisory control and data acquisition (SCADA). Related works and the contribution of the present article are addressed in the beneath:

A. Related Works

References [1] and [2] are among the former researchers in the area of water network anomaly detection. They apply a clustering algorithm that has earlier been successfully applied to n-dimensional data spaces on the internet for real-time anomaly detection in water management systems. They describe the evaluation of the anomaly detection software when integrated into a wired SCADA system. Since the main focus of the present paper is the anomaly detection in the smart water networks by using wireless platforms, some previous related works are evaluated in terms of the platform and communication aspects in the following: [3] discussed the anomaly detection in the water networks by using a WSN. They used Intel Mote based sensor nodes (SNs) to collect the measured data and transmitted the Min/Max/Average values via a GPRS modem to a back-end server. Although many researchers have followed this methodology, the main disadvantage of this approach is that all aggregated data have to be transmitted to the back-end system for analysis, which yields to deplete the battery-powered nodes. Besides, Min/Max/Average values are not enough for accurate anomaly detection in many cases. Further, it cannot work with the medium and large-sized networks because of the data collision issues. To compensate for the energy-wise weakness of WSNs, [4] and [5] dealt with a WSN based fault detection by using the Intel mote (Imote2). They introduced an energy-aware methodology called Floating Input Approach (FIA) to detect about 20 types of sensor and system faults in a closed space environment while saving more than 50% of consuming electrical power for each node in another research reported by [6]. However, the platform they used could only cover a short Communication range and that they also used to transmit all the gathered data directly to a cluster-head for the model making and energy management. Recent methodologies of anomaly detection have also been introduced in [7], [8], and [9]. A literature survey focusing primarily on machine learning-based anomaly detection is also given in [10]. As a part of the network-based anomaly detection systems, the WSN gateways are employed to provide WSN access to the hosts of wide-area networks. To design appropriate gateways, authors in [11] present a web-based

Manuscript received January 3, 2020; revised February 1, 2020; accepted February 23, 2020. Date of publication February 28, 2020; date of current version May 15, 2020. The associate editor coordinating the review of this article and approving it for publication was Dr. Prosanta Gope.

The author is with the Control Systems Group, Faculty of Engineering, University of Zanjan, Zanjan 38791-45371, Iran (e-mail: mebab@znu.ac.ir).

Digital Object Identifier 10.1109/JSEN.2020.2976650

system for managing and querying sensor networks with the WSNs having a 3-level regional hierarchy. Reference [12] presents a modular WSN gateway design to be deployed in heterogeneous network environments and enables remote upgrading. Since the varied wide-area networks may require connectivity with the same WSN, they develop a WSN gateway design with a configurable engine for protocol translation. Reference [13] outlines the system design of a gateway that meets the hardware and software demands of data gathering networks and presents a prototype implementation of this design using the popular star-gate platform. The gateway communicates with the back-end servers through a periodic query-response protocol, which is robust to connectivity outages. Commands from the back-end servers instruct the gateway to wake up the sensor network, perform bulk data downloads and finally upload the collected measurements to a remote server. They investigate mechanisms to minimize energy consumption at the gateway including completely powering it off in between query intervals and batching data uploads. While they ground their design using the 'Stargate' device and evaluate its power consumption using different long-haul radios, they present several platforms that could become viable alternatives in terms of their cost/performance ratio. Another work reported by [14] presents the development of a general gateway system for data collection in WSNs. The unique characteristic of the developed WSN gateway is the easily configurable design for nearly any WSN data collection applications with diverse data management and WSN protocols where it supports the WSN transport layer at the gateway. Their implementation leverages Java and widely adopted TinyOS platform for various WSNs. Nevertheless, some of the mentioned articles refer to the sensor-side fault detection with the lack of detailed design information. Besides, the articles in the area of the gateway design mainly focus on the general connectivity issues and [15] is the only work found to address the sensor-side in detail.

B. Main Contribution

Edge-based anomaly detection is made of the coordinating center and sensor-side algorithms. Despite the addressed references, the main contribution of the present article is to focus on the Edge-based anomaly detection platform either in the center or sensor-side. It mainly addresses different design-related challenges in the edge-processing based platform but not the theoretical aspects of anomaly detection. The corresponding data preparation is represented and the network management is also discussed in detail. The primary stage of this work was already published in [16] with the focus on the edge processing utilization in the anomaly detection and the corresponding sensor-side design methodologies. Now, it is extended to the center-side design with further improvements towards the autonomous operation of the implemented LoRa-based detection system. Authors in [17] focus on anomaly detection by using the CR. Besides, [15] and [16] represent that to handle fast measurements with the slow TX rate as is the case with the LoRa technology, the edge processing makes it possible to optimally utilize the sensor-side resources on a single-core processor. In the present work, not only the idea in [17] is implemented with the core technical details, but also it is combined with the network management functionalities both in the center and sensor-side.

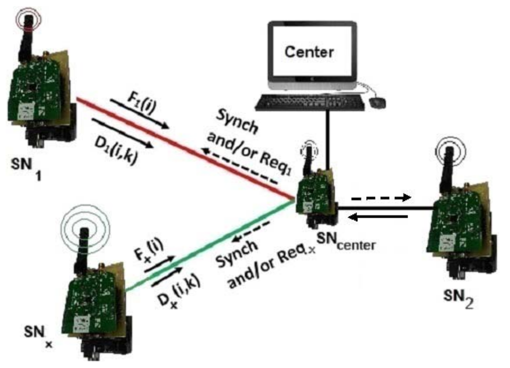


Fig. 1. LoRa based SNs and center in communication.

The rest of the paper is organized into a few main sections as follows: The next section II summarizes the system characteristics and addresses the practical limitations. The primary definitions and the desired structures are also described in this section. Then, the sensor-side requirements are categorized in section III and the proposed methodologies are presented. The Center-side design in section IV focuses on the other side of the implementation and discusses the proposed network management. Section V exhibits the test results of the edge-based anomaly detection in a worst-case scenario. Ultimately, section VI concludes the article in brief.

II. System Overview and Limitations

Fig. 1 exhibits a star-based, node-to-node scheme implemented for the proposed wireless anomaly detection. All nodes and the center have been developed on the single-core, 32 MHz processor Arduino/Genuino 101. The SNs (SN_1, SN_2, $\ldots SN_x$) can be located up to a few kilometers away from each other or from the central node. Anomalies are detected in two consecutive stages: after an event is primarily detected in the sensor-side, the secondary center-side evaluation carries out based on the received relevant information. The local SNs compress their real-time measurements, store them in their own SD memory card, and calculate the data compression rate (CR). The SNs primarily detect potential events in real-time by evaluating a filtered compression rate (FCR) as disclosed in [16]. To optimize the detection system, each SN appoints its worst event among all occurred events (if any) during a predefined period. Hence, each detective node that is called $'source\ node'$ in this article, notifies the center about the worst event of that period with an appropriate timestamp and resumes the next round. If the center asks for complementary data linked to that timestamp, the source node will look back for the relevant pre-fault compressed data stored on the SD memory card. Then, some sort of the pre-fault compressed data packets will be transmitted to the center for the final comprehensive evaluation. Dealing with the compressed data guarantees the use of a minimum occupied storage space on edge SNs, lowers both the communication time on-air and the congestion probability and thereafter, minimizes resultant power consumption within the network. The following limitations affect implementation:

- Available static random access memory (SRAM), speed, and power consumption have to be considered in the design of the algorithms. Keeping with a fixed sensor rate limits the single-core implementation where all tasks have to be finalized during a fixed loop time. Thereafter, data Communication cannot use acknowledgement (ACK) mode due to its time-consuming nature.
- The low TX/RX rate (0.018 - 37.5 kbps) associated with the LoRa technology, the 1% duty cycle rules, and the number of SNs limits the maximum TX time of each SN, particularly on LoRaWAN platforms.
- Utilizing a WSN in urban areas confines the reliable Communication range and increases the data packet loss probability. Resolving these issues introduces more complexity to both the center and sensor-side designs which are considered in the algorithms.

III. Sensor-Side Design

The SN in the proposed system means to measure the physical parameters, evaluate the measurements to detect suspicious events and communicate back and forth with a center. The parameters should be sampled at the maximum achievable fixed rate which is limited by the developed hardware and algorithms. The next subsection provides design considerations.

A. Sensor-Side Design Considerations

The size of WSN, power, and available computing resources affect the sensor-side design. For a single-core based SN, there might be two major design methodologies as follows:

- Fixed loop time- As implemented in the present research, all the operations including measurements, filtering, data compression, recording, calculations, and TX/RX have to be carried out during a fixed time.
  - Advantages: there will be a fixed time interval between each measured data and one can count on this sampling time while evaluating the decompressed data in the center-side. The loop time is limited from up and down by the SRAM size, sampling time, and the number and duration of tasks in the loop (see [16] for more details).
  - Disadvantages: The maximum number of SNs is limited in this scheme. Because in the worst-case scenario all the SNs communicate their own worst event back to the center. However, according to [18], to avoid any concurrency, the channel activity detection (CAD) mechanism has to be applied. In this case, more SNs than permitted might not have the chance to transmit during a fixed loop time and the number of packet loss may increase. Further, it is not possible to use TX/RX with acknowledgment in this scenario. Therefrom, there is no guaranty for the whole packet reception. The maximum supported number of SNs that can transmit concurrent detections depends on the following items:
    1. sensor-side loop time (T_L= 10 s)
    2. Notification's transmission time (Δt)- This depends on the chosen LoRa mode, ISM band, and the time on-air for the notifications. The LoRa mode determines the spreading factor (SF), bandwidth (BW), cyclic redundancy check (CRC) and other settings of the module. With the Asian ISM band 433 MHz, the first mode 1 represents Δt= 1570 ms, mode 5 with SF= 7 and CRC= 1 gives Δt= 437 ms, and mode 10 offers Δt= 266 ms to transmit for instance ($E003000102192155$) in a short distance. With the selected mode 5 with BW= 125 kHz, the maximum possible number of the SNs (n) handling simultaneous event detection can be approximated by (1). It represents a rare case where all SNs want to communicate concurrently without packet loss that is unlikely in a wide range LoRa network. Then, the real number of SNs can be even more.

$$n = \frac{T_L}{\Delta t} = \frac{10}{0.437} \approx 22 \tag{1}$$

- Variable loop time- In this scenario, the sampling rate of the measurement is variable. Every source node waits for a free channel as long as required and then transmits. If the number of source nodes increases, the waiting time for them to successfully transmit can be longer as well.
  - Advantages: There might be more successful communications due to activating the LoRa ACK mode.
  - Disadvantages: There won't be a fixed measurement sampling rate. If edge processing applies where data is compressed, there won't be precise times linked to the decompressed data. In consequence, signal processing won't be as precise as the former method (fixed loop time). If a longer time is needed for a TX, the compression data array will be overwritten at the end of a loop execution which results in lower performance in the central-based anomaly detection.

Algorithm 1 and Algorithm 2 illustrate simplified Pseudo Codes for the mentioned fixed loop time scenario where sensor-side tasks are implemented sequentially in a main loop. Running an interrupt service routine alongside the main loop execution, the measurements are sampled and pre-filtered in real-time with the sensing rate T_s = 10 ms. As a result, the input-arrays for compression are made in either of the following positions in the main loop:

- Stage 1 (before compression): As shown in Algorithm 1, an input-array is filled up with the pre-filtered data within an internal waiting loop for feeding the algorithm.
- Stage 2 (after compression): Only a part of the next required input-array for compression is filled up with the pre-filtered data before running the next waiting loop. The number of the entered data into the input-array must be lower than the input-array size in this stage, meaning that the relation (2) has to be fulfilled.

$$T_{tasks} < P_x^i \times T_s \tag{2}$$

where T_{tasks} is the total time of executing the tasks during the main loop execution time, $P_x^i = 1000\ bytes$ is the fixed size of the x^{th} input-array associated with the

Algorithm 1 Pseudo Code for the Main Loop of SN

```
Data: Input data, arrays and Interrupt Timer settings
while do
    while input array is not complete do
        Call Algorithm 2 every $T_s$= 10 $ms$;
    end
    Call Comp. Algorithm; Determine CR; Apply a low
    pass filter to CR;
    if received any synchronization Request then
        Update RTC and other Sensor-side settings;
    end
    Read RTC;  timestamp = RTC time;
    if Anomaly exists or it is inside timer1 period then
        Start timer1 only by the first anomaly;
        Count time until timer1 = 1 $min$;
        Verify the worst anomaly while timer1< 1 $min$;
        if timer1 $\geq$ 1min then
            Notify Center the worst anomaly's timestamp;
            Reset timer1 and Start timer2;
        end
    end
    Append timestamp and data packet to SD card;
    if A Data Request exists and $T_{timer2} \leq 1\ min$ then
        Read SD card and transmit data;
        Reset timer2;
    end
end
```

Algorithm 2 Pseudo Code for Callback Function

```
Reading Sensor data;
Scale measured value; Apply pre-filter;
Enter new data in the input data array;
Back to where interrupt started in Algorithm 1;
```

i^{th} loop execution. Each data element in the array is a $1-byte$ integer number (0 to 255) and $x = 1, \ldots, P(i)$ where $P(i)$ is the total number of the input-arrays related to a specific period of measurement. The compression rate (CR) is obtained as:

$$CR\% = 100 \times [\frac{P_x^i - C_x^i}{P_x^i}] \tag{3}$$

where C_x^i is the size of the x^{th} compressed array, associated with P_x^i. The compression method is discussed more in [15] and [16]. The present research follows the edge-based methodology implemented in [15] where the local anomaly detection is carried out primarily by each SN based on the comparison of the Comp. Rate value with a fixed or adaptive threshold as reported by [19]. According to Fig. 2, an appropriate timestamp appends to each compressed array (CA) and the result is stored in an SD memory card. The compressed array is stored instead of the normal array to take lower memory space where $\{\forall x, i\ :\ C_x^i < P_x^i\}$. The CR is evaluated to catch the suspicious events with a suitable methodology such as either an adaptive threshold violation criteria [19] or a fixed threshold

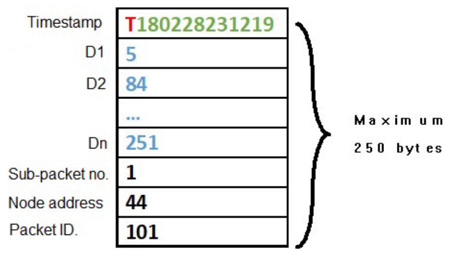


Fig. 2. An example of a sub-packet data format on SD memory card.

methodology proposed by [15]. Upon detection of an event, a timer is activated and the worst event detection process carries out in a certain period during the next loop's execution. Detecting the worst event, the source node (SN_x) transmits an E-type notification $(E_x(i))$ which is the i^{th} worst event when there is a free channel according to the LoRa CAD mode, excluding the stranger WSNs operating in the neighborhood. There might be instances where some SNs for example, SN_1 and SN_2 in Fig. 1 cannot reach each other. Then, the simultaneous detection of the event by those out of range SNs causes a little risk of their notification loss in the center. An event notification is a timestamp including the SN number, the time of the worst event detection, and an event identification symbol (E). As an example, $E011180324162930$ represents an E-type notification by the SN number: 011 detected in the date: $2018/03/24$ at $16:29':30''$.

By receiving a data query $(R_x(i))$ from the center, the SN_x primarily stores the query. Then, having the relevant starting and stopping address of the pre-fault data on the SD memory card, it makes appropriate sub-packets and transmits to the center successively. It is noteworthy that there won't be any concurrent TX in the WSN since the SNs receive the data query one by one with enough delays in between.

To minimize the risk of concurrent E-type notification loss, the SNs should transmit the notifications according to the CAD mode in $SX1276$, though there still might be a little risk of:

- The D-type or E-type packet loss where some SNs are out of range.
- The center is issuing a repetitive data query, while the source node intends to transmit E-type notifications. To ignore this concurrency, The SN waits for a specific period until the data query is terminated. The related notification can be transmitted afterward.

A synchronization process carries out individually to ensure the last alive or joint SNs have the right clock time. The new joining SNs are synchronized before they start operations. Otherwise, their data will be useless.

The sensor-side Algorithm 1 indicates a single loop where all the calculations are done sequentially and some tasks are more time-consuming than the others. The loop might be busy for instance with TX when a command signal receives from the center. Then, the SN will not be able to receive it. The implemented solution in the center-side is, to broadcast command signals for a longer period than the SN's main loop execution time (10 s) so that the SN gets a chance to detect at least one of them while running its own tasks.

B. Sensor-Side Data Packet Configuration

Depends on the data variation within the fixed-size input-array (1 kB), the compression algorithm produces the related compressed array (CA) with the following size:

$$32\ bytes \leqslant C_x^i \leqslant 1130\ bytes \tag{4}$$

where 32 $bytes$ compressed array size obtains experimentally when fixed numbers constantly fill out the input-array.

A few minutes of the compressed data includes several kilobytes of the combination of the data and timestamps stored in an SD memory card. It cannot be fully transmitted to the center since the maximum permitted packet size for TX by LoRa is limited to 250 $bytes$. Therefore, the long CAs are divided into several sub-packets each maximum in 250 $bytes$ for TX, where the first sub-packet of each CA starts with a timestamp. In brief, for the i^{th} query, $P(i)$ number of 1 kB pre-filtered data generates $P(i)$ number of CAs with size C_x^i. For example, the CA_x^i generates $S_{P(i)}^i$ sub-packets (SP) that start from $SP_{(x,1)}^i$ to $SP_{(x,S_{P(i)}^i)}^i$. In order to transmit a few minutes of pre-fault compressed data as it is requested by the center, the SN has to send too many sub-packets, each in every main loop execution. Every requested minutes of CAs have their own total lengths as depicted in (5).

$$Size\ of\ CA_x^i = \sum_{x=1}^{P(i)} C_x^i \tag{5}$$

Since timestamps and ID numbers are added to sub-packets, the following inequality exists in (6):

$$\sum_{n=1}^{S_n^i} SP_{(x,n)}^i > CA_x^i \tag{6}$$

The received sub-packets in the center have been signed by the originated SNs before transmission as follows:

- ID = 100: for the first sub-packet of a long packet with a timestamp.
- ID = 101: for a single packet shorter than permitted payload size (250 $bytes$) starting with a timestamp where no need to make any sub-packet.
- ID = 102: for the last sub-packet of a final packet with a timestamp.
- ID = 103: for the middle sub-packets of a long packet without a timestamp.
- ID = 104: for the last sub-packet of a long packet without a timestamp.
- ID = 105: for the last sub-packet of a final packet without a timestamp.

Fig. 3 illustrates a sub-packet format with/without a timestamp and Fig. 4 denotes an example of a data packet among other packets for communication.

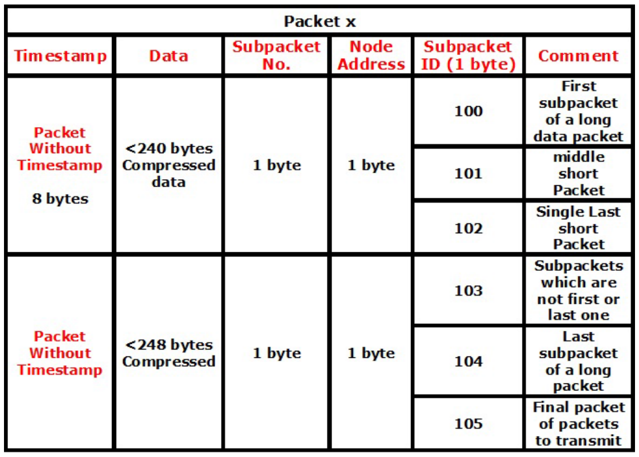
IV. Center-Side Design

A. Center-Side Design Considerations

Center detects anomalies conducting following tasks:

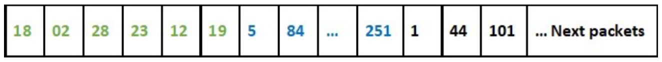
- Commands: Center issues different types of commands for numbering new joined SNs, different types of synchronization, data query, and/or settings up SNs according to a predefined plan as:
  - Numbering: This is to assign an exclusive number to each unidentified SN.
  - Synchronization (sync): The $sync$ command is issued in two cases:
    - Coming a new joining SN, center issues a $sync$ command to that SN and deactivates its notification history in the center.
    - If the center starts up, it broadcasts $sync$ commands to all SNs.

    In any case, the current clock $'time-date'$ in the center is transmitted together with other settings for a while to ensure the target SN/SNs get it. The setting includes $'clear/unclear'$ command for SD cards.
  - Data query: The center makes a queue from received event notifications. They include the source node number and the time of their detected event. To guaranty the minimum D-type packet loss, any new data query has to be issued after the previous packet receives in full. In case of packet loss, a limited number of inquiries issue to the same SN to overcome the loss issue. There might be some source nodes where they fail after they notify the center. These SNs are extracted if a query-wait time for data (t_{WD}) elapses. This will prevent frequent Communications to those failed SNs and lowers the additional power consumption.
- Reconstruction: The center means to reconstruct original compressed packets from the received sub-packets. Provided that there is no sub-packet loss, each pure compressed packet is then decompressed to be used for final anomaly detection.



Packet x					
Timestamp	Data	Subpacket No.	Node Address	Subpacket ID (1 byte)	Comment
Packet Without Timestamp 8 bytes	<240 bytes Compressed data	1 byte	1 byte	100	First subpacket of a long data packet
				101	middle short Packet
				102	Single Last short Packet
Packet Without Timestamp	<248 bytes Compressed	1 byte	1 byte	103	Subpackets which are not first or last one
				104	Last subpacket of a long packet
				105	Final packet of packets to transmit

Fig. 3. Single packet or a sub-packet with/without timestamp.



18	02	28	23	12	19	5	84	...	251	1	44	101	... Next packets

Fig. 4. Example of a sub-packet including timestamp, data, sub-packet number, node number and sub-packet identification (ID) for TX.

B. Center-Side Data Management

Algorithm 3 shows the pseudo-code, showing the center-side operations. It is divided into three main loops. The first loop, E-Loop evaluates received event notifications and D-Loop performs data packet management. The third small

Algorithm 3 Pseudo Code for Center-Side Node

```
Data: Set up input data, define sync type
if Center is turned on then
    Broadcast defined sync command to all SNs;
end
while true do
    if a packet received or there is an interrupt then
        if SN number is correct then
            if received data is E-Type then
                1. Store received Timestamps in a Queue;
                2. Check Data reception as well;
                3. Check packet loss;
                4. Apply management time settings;
                5. Update required variables;
                if An SN exists in the queue or repetitive
                Query is needed for the same SN then
                    Send Data Query to that SN;
                end
            end
            else if received data is D-Type then
                1. Decode received data sub-packet;
                2. Detect packet loss and update Packet loss
                number if any;
                3. Send Data Query in time and if needed;
                if End of sub-Packets of a packet and no
                packet loss then
                    1. concatenate sub-packets and make
                    original packet;
                    2. Decompress full packet;
                end
                if End of all packets and No packet loss then
                    if An SN exists in the queue or repetitive
                    Query is needed for the same SN then
                        Send Data Query to that SN;
                    end
                end
            end
            else if received data is B-Type then
                Send sync to targeted SN;
            end
        end
    end
end
```

loop deals with the synchronization task. Within the D-Loop, it may switch between either of TX or RX modes depending on the task it is performing. However, during the E-Loop, it will only stay in RX mode.

A new query for data issues only if the previous one has been received successfully or if there is a packet loss so that repeating a query to the same SN is needed to improve the robustness of the anomaly detection. While in RX mode, as soon as the LoRa in the center-side detects a received payload, it is decoded in order to distinguish either it is a D-type, E-type or B-type to jump on the related loop. An event notification array (CE-Array) with two pointers controls the Communications within the WSN. This array stores received notifications from source nodes in the form of the timestamp. The first pointer allocates the new coming notifications in the array and the second one determines SNs to be communicated with. Arduino101 has 24 kB SRAM memory which is used by many variables, particularly in the sensor-side. The rest can be assigned to this array. This is why the array is of limited size and its items might be overwritten when the array gets full. Targeting a specific SN based on a timestamp selection process in the center-side, the compressed data, already stored in the SD card memory of that SN has to be transmitted back to the center. In case of no packet loss, that timestamp gets stamped in the array and CE-Array is updated accordingly.

The following tasks carried out in the center:

- Numbering SNs if set as 'Automatic numbering' versus 'Manual numbering'.
- Diagnosing the received packet types and IDs.
- Synchronizing either one SN due to an individual request for the synchronization or all SNs according to a predefined start-up plan.
- Evaluation of packet loss and perform needed activities.
- Updating the CE-Array according to either the received E-type notifications or successfully communicated ones.
- Receiving data packets and extract timestamps from sub-packets and concatenate them to make full compressed packets of size 1000 bytes.
- Carrying out a limited number of query-wait actions in case of a packet loss.
- Decompressing the concatenated compressed full packets of each SN.
- Ignoring unauthorized SNs and excluding unresponsive ones. Also acknowledgment and securing newly received notifications and/or data sub-packets while the execution of the other tasks.

C. Center-Side Timing Diagram and Limitations

Fig. 5 illustrates the timing diagram of the operations in the center-side through A_1 to A_3 as follows: The uppermost diagram A_1 exemplifies the worst event notifications E_1, E_3 and E_5 issued by the SN_1 and E_2, E_4 and E_6 issued by the SN_2 respectively. After notifications receive at the center with a small delay, the timestamp of the source nodes (SN_1 and SN_2) are primarily stored in the CE-Array. According to diagram A_2, the first compressed data query associated with E_1 transmits repeatedly for a fixed period (T_R) to ensure the first source node (SN_1) receives at least one of them during its loop execution. The diagram A_3 denotes the operations in SN_1 accordingly where it performs two important functions as follows: as soon as SN_1 receives the query (at t_0), it waits for a predefined waiting time ($T_w > T_R$) and then starts to transmit the sub-packets one by one ($SP_1, SP_2, \ldots SP_5$). This waiting time is to ensure T_R is elapsed and the center can act as the receiver of the data packets. This will, in turn, lower the probability of packet loss. After the last data packet associated to E_1 receives at the center, the second data query issues to SN_2 at t_2 (compare A_2 and A_3). The new query asks data associated to E_2 from SN_2. According to A_3, SN_1 can transmit sub-packets, catch new

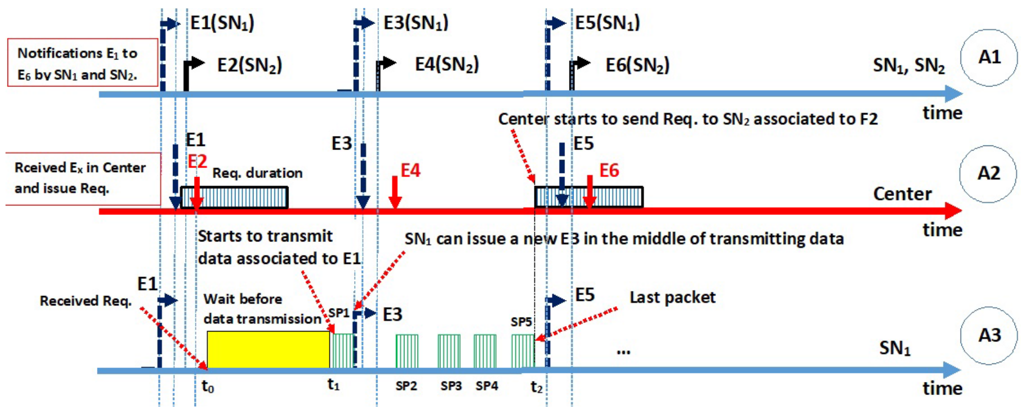

E1(SN₁)
E2(SN₂)
E3(SN₁)
E4(SN₂)
E5(SN₁)
E6(SN₂)
Notifications E₁ to E₆ by SN₁ and SN₂.
SN₁, SN₂
A1
time
Center starts to send Req. to SN₂ associated to F2
Rceived Eₓ in Center and issue Req.
E1
E2
Req. duration
E3
E4
E5
E6
Center
A2
time
Starts to transmit data associated to E1
SN₁ can issue a new E3 in the middle of transmitting data
Received Req.
E1
Wait before data transmission
SP1
E3
SP5
E5
Last packet
...
SN₁
A3
time
t₀
t₁
SP2
SP3
SP4
t₂

Fig. 5. Timing diagram for the center operation.

events in between, and transmit the corresponding notifications (for example E_3).

D. Dynamic Time Scheduling by Using LoRa CAD Mode

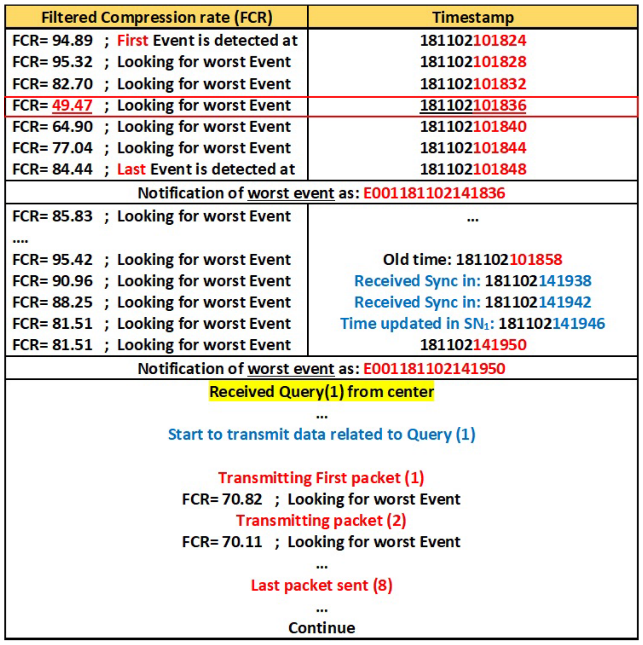
While SN_x runs its fixed-time main loop, there is only one chance to prepare and transmit a periodical worst event notification ($E_x(i)$) and/or a data sub-packet. To that end, concurrency is likely to happen in a network of several SNs transmitting their notifications or data sub-packets. To lower the chance of concurrency, the CAD mode is activated for a short while to detect a free channel for transmission. Running this mode, there might be different scenarios as follows:

- SNs come to check the channel one by one and a free channel is detected by the first SN to transmit: Receiver detects this packet with the highest probability with no packet loss. Other SNs may not get a chance to transmit if they run CAD mode immediately after the first SN. To overcome this issue, CAD mode has been re-activated in an internal waiting loop for a limited time until a free channel is detected for other SNs. If there are many SNs in the immediate-to-transmission status, there will be a possibility of packet loss anyway.
- A free channel is detected simultaneously by more than one SNs and all transmit concurrently: There is less chance for a simple receiver like what is used in this research to catch all and packet loss is likely to happen. If this happens ones, it might be repeated in the next loop executions. To avoid this, a random delay time has been added to each SN to differ its activation time of CAD mode in the next periods.

Above mentioned methodologies result in a dynamic localization of the packets in a queue during each loop time to utilize the maximum possible free time for TX and to minimize the data packet loss by reducing the concurrency.

V. Experimental Results

An experiment has been organized for plausibility check and to reflect how the SNs detect odd situations and communicate with the center properly within an indoor laboratory environment. For the sake of simplicity of representation, the center manages two SNs (001 and 002) as shown in Fig. 6 and Fig. 7 where only SN_1 among two SNs prints out the results. Thereafter, the following operations carries out one by one:



Filtered Compression rate (FCR)	Timestamp
FCR= 94.89 ; First Event is detected at	181102101824
FCR= 95.32 ; Looking for worst Event	181102101828
FCR= 82.70 ; Looking for worst Event	181102101832
FCR= 49.47 ; Looking for worst Event	181102101836
FCR= 64.90 ; Looking for worst Event	181102101840
FCR= 77.04 ; Looking for worst Event	181102101844
FCR= 84.44 ; Last Event is detected at	181102101848
Notification of worst event as: E001181102141836	
FCR= 85.83 ; Looking for worst Event	...
....	
FCR= 95.42 ; Looking for worst Event	Old time: 181102101858
FCR= 90.96 ; Looking for worst Event	Received Sync in: 181102141938
FCR= 88.25 ; Looking for worst Event	Received Sync in: 181102141942
FCR= 81.51 ; Looking for worst Event	Time updated in SN₁: 181102141946
FCR= 81.51 ; Looking for worst Event	181102141950
Notification of worst event as: E001181102141950	

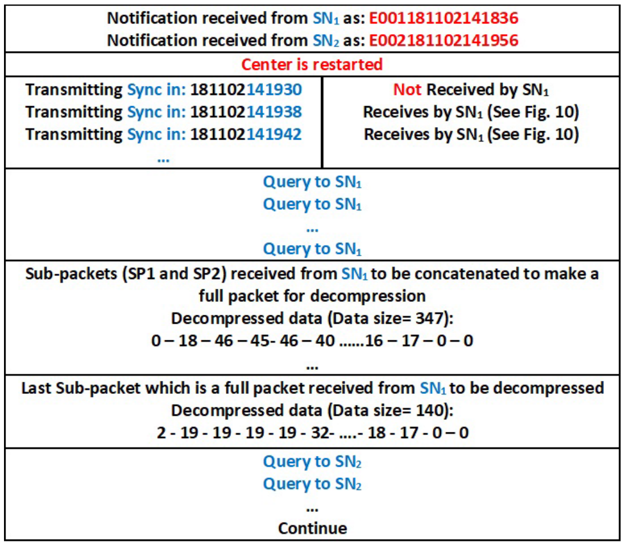
```
Received Query(1) from center
...
Start to transmit data related to Query (1)

Transmitting First packet (1)
FCR= 70.82  ; Looking for worst Event
Transmitting packet (2)
FCR= 70.11  ; Looking for worst Event
...
Last packet sent (8)
...
Continue
```

Fig. 6. Real output in the sensor-side.

Both SNs detect repetitive events every 10 s as if they are working in the worst-case scenario and notify the center of the worst ones after 1 min waiting for the worst event. The worst event detected by SN_1 is the one with the lowest $FCR = 49.47$ as marked with the red colored rectangular. The worst event is transmitted to the center by the timestamp 181102101836. It is also indicated in the center-side with the same timestamp. In addition the center receives another notification from SN_2.

Before a query (request) is transmitted to SN_1 to ask for the corresponding compressed data, the center is restarted accidentally. Then, it broadcasts synchronization commands repeatedly for a while. The SNs receive at least one of them and get synchronized accordingly with the last real clock time of the center. Now, the center transmits query to SN_1. Since the Communication with the acknowledgment cannot be used in the fixed loop scenario in this research (please see section II), the queries have to be transmitted repeatedly at least for one loop execution time. The SN_1 receives one of them and starts to transmit related sub-packets of the pre-fault.

Fig.6 illustrates that all the normal activities of the SNs continue while they are communicating with the center.



Notification received from SN_1 as: E001181102141836
Notification received from SN_2 as: E002181102141956

Center is restarted

Transmitting Sync in: 181102141930	Not Received by SN_1
Transmitting Sync in: 181102141938	Receives by SN_1 (See Fig. 10)
Transmitting Sync in: 181102141942	Receives by SN_1 (See Fig. 10)
...	

Query to SN_1
Query to SN_1
...
Query to SN_1

Sub-packets (SP1 and SP2) received from SN_1 to be concatenated to make a full packet for decompression
Decompressed data (Data size= 347):
0 – 18 – 46 – 45- 46 – 40 ……16 – 17 – 0 – 0
...

Last Sub-packet which is a full packet received from SN_1 to be decompressed
Decompressed data (Data size= 140):
2 - 19 - 19 - 19 - 19 - 32- ….- 18 - 17 - 0 – 0

Query to SN_2
Query to SN_2
...
Continue

Fig. 7. Real output in the center-side.

For example, the worst event detection continues while transmitting data sub-packets. After all of the sub-packets that make full packets receive at the center, they are concatenated properly and full packets are prepared for the decompression.

The center transmits data queries according to the source nodes in the queue (SN_1 and then SN_2). Therefore, after the final packet receives from SN_1, the center transmits the data request to SN_2 and waits for the sub-packets of anomaly-related data to be arrived. Other notifications will be managed by the center in the meantime even. After all the pre-fault packets received from source nodes are decompressed, the original filtered data of SNs is available in the center and any kind of anomaly detection can be carried out. In addition to real-time data evaluation, they can be stored in a database for the future uses. From this point forward, data can be available through internet for authorized ones to make Internet of Things (IoT) from the proposed LoRa-based platform.

VI. Conclusion

An autonomous LoRa-based anomaly detection, applicable to a variety of network-based systems implemented and described in detail. To serve fixed sensor-side sampling rate and effective data transmission within the network, requirements figured out and coordinated algorithms developed both in the center and sensor-side. Some approaches developed to optimize the system operation under the worst-case scenario where most concurrency could occur. The channel activity detection (CAD) mechanism customized to localize different sensor nodes in an effective time-based frame, facilitating Communication with the minimum packet loss.

References

[1] M. Raciti, J. Cucurull, and S. Nadjm-Tehrani, "Anomaly detection in water management systems," in *Critical Infrastructure Protection*. Berlin, Germany: Springer, 2012, pp. 98–119.

[2] D. Loureiro, C. Amado, A. Martins, D. Vitorino, A. Mamade, and S. T. Coelho, "Water distribution systems flow monitoring and anomalous event detection: A practical approach," *Urban Water J.*, vol. 13, no. 3, pp. 242–252, Jan. 2015.

[3] I. Stoianov, L. Nachman, S. Madden, and T. Tokmouline, "PIPENETa wireless sensor network for pipeline monitoring," in *Proc. 6th Int. Conf. Inf. Process. Sensor Netw. (IPSN)*, 2007, pp. 264–273.

[4] M. Babazadeh, H. Kreowski, and W. Lang, "Selective predictors of environmental parameters in wireless sensor networks," *Int. J. Math. Models Methods Appl. Sci.*, vol. 2, no. 3, pp. 355–363, 2008.

[5] M. Babazadeh and W. Lang, "Fault diagnosis while monitoring environmental variables by a sensor network," *IFAC Proc. Volumes*, vol. 42, no. 13, pp. 272–277, 2009.

[6] M. Babazadeh, I. Lal, and W. Lang, "Energy saving by using floating input approach in a wireless sensor network," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2009, pp. 595–599.

[7] B. Shi, P. Wang, J. Jiang, and R. Liu, "Applying high-frequency surrogate measurements and a wavelet-ANN model to provide early warnings of rapid surface water quality anomalies," *Sci. Total Environ.*, vols. 610–611, pp. 1390–1399, Jan. 2018.

[8] L. Millán-Roures, I. Epifanio, and V. Martínez, "Detection of anomalies in water networks by functional data analysis," *Math. Problems Eng.*, vol. 2018, pp. 1–13, Jun. 2018.

[9] H. R. Ghaeini, D. Antonioli, F. Brasser, A.-R. Sadeghi, and N. O. Tippenhauer, "State-aware anomaly detection for industrial control systems," in *Proc. 33rd Annu. ACM Symp. Appl. Comput. (SAC)*, 2018, pp. 1620–1628.

[10] D. Ramotsoela, A. Abu-Mahfouz, and G. Hancke, "A survey of anomaly detection in industrial wireless sensor networks with critical water system infrastructure as a case study," *Sensors*, vol. 18, no. 8, p. 2491, Aug. 2018.

[11] K.-I. Hwang, J. In, N. Park, and D.-S. Eom, "A design and implementation of wireless sensor gateway for efficient querying and managing through world wide Web," *IEEE Trans. Consum. Electron.*, vol. 49, no. 4, pp. 1090–1097, Nov. 2003.

[12] L. Wu, J. Riihijarvi, and P. Mahonen, "A modular wireless sensor network gateway design," in *Proc. 2nd Int. Conf. Commun. Netw. China*, Aug. 2007, pp. 882–886.

[13] R. Musaloiu-E, R. Musaloiu-E, and A. Terzis, "Gateway design for data gathering sensor networks," in *Proc. 5th Annu. IEEE Commun. Soc. Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, Jun. 2008, pp. 296–304.

[14] N. Erratt and Y. Liang, "The design and implementation of a general WSN gateway for data collection," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 4392–4397.

[15] M. Babazadeh, S. Kartakis, and J. A. McCann, "Highly-distributed sensor processing using IoT for critical infrastructure monitoring," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, Dec. 2017, pp. 1065–1074.

[16] M. Babazadeh, "Edge analytics for anomaly detection in water networks by an Arduino101-LoRa based WSN," *ISA Trans.*, vol. 92, pp. 273–285, Sep. 2019.

[17] S. Kartakis and J. A. McCann, "Real-time edge analytics for cyber physical systems using compression rates," in *Proc. 11th Int. Conf. Auton. Comput. (ICAC)*, 2014, pp. 153–159.

[18] C. Pham, "Investigating and experimenting CSMA channel access mechanisms for LoRa IoT networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[19] S. Kartakis, W. Yu, R. Akhavan, and J. A. McCann, "Adaptive edge analytics for distributed networked control of water systems," in *Proc. IEEE 1st Int. Conf. Internet-Things Design Implement. (IoTDI)*, Apr. 2016, pp. 72–82.

Mehrdad Babazadeh received the B.Sc. degree in power systems from the Iran University of Science and Technology, Tehran, Iran, in 1992, the M.Sc. degree in control engineering from the University of Tehran, Tehran, in 2000, and the Ph.D. degree in control engineering from the University of Bremen, Bremen, Germany, in 2010. From 2010 to 2012, he worked as a Wind Power Plant Control Lead Engineer at Vestas, Denmark. Since 2012, he has been an Assistant Professor with the Control Systems Group, University of Zanjan, Zanjan, Iran. In addition, he did his postdoctoral research on the implementation of anomaly detection in smart water networks at the Imperial College London, London, U.K., from 2016 to 2017.