A Framework for Optimal Fault Tolerance Protocol Selection Using Fuzzy Logic on IoT Sensor Layer

Mehdi Nazari Cheraghlou
Department of Computer Engineering
South Tehran Branch, Islamic Azad University, Tehran, Iran
St_m_NazariCheraghlou@azad.ac.ir

Ahmad Khadem-Zadeh[*]
Iran Telecommunication Research Center (ITRC)
Tehran, Iran
zadeh@itrc.ac.ir

Majid Haghparast
Department of Computer Engineering
Yadegar-e-Imam Khomeini(RAH) Shahre Rey Branch, Islamic Azad University, Tehran, Iran
haghparast@srbiau.ac.ir

Received: 25 December, 2017 - Accepted: 12 February, 2018

Abstract— **Internet of things (IoT) is the intersection of many different technologies and networks. The management of these is very complex and important. One of the dimensions that must be considered in IoT management is the fault tolerance level of the systems. Of particular importance is the fact that lack of, or a weakness of this feature would threaten the survival and existence of the system.**

In this paper, a framework called FISWSN is proposed to evaluate the fault tolerance feature of protocols and frameworks in the first layer of the IoT architecture (Sensing layer). Given that the Fault Tolerance is a qualitative parameter, the linguistic variables and fuzzy logic are used to implement the proposed framework. FISWSN is used to elect optimal protocols and architectures with the aim of increasing fault tolerance and higher efficiency in the final output but at a lower cost.

Keywords- Internet of Things (IoT), Wireless Sensor Network (WSN), Fault Tolerance, Cloud Computing, Fuzzy Logic.

I. Introduction

The progress and transformation of the Internet has been such that at the beginning it has only included content. In the second stage the Internet service has been proposed. Then the Internet of People came into the world and examples of which are a variety of social networks. Internet of things (IoT) is the current topic under study. In IoT, the various constituent elements must be able to preserve their own independent characteristics and functions and work together in a manner that they are fully integrated and managed under a single architecture. So far, different architectures are presented including a three-tier architecture, an SOA - Based architecture and a five-layer architecture. All of them are described in [1]. The most common IoT architecture is presented in Fig. 1.

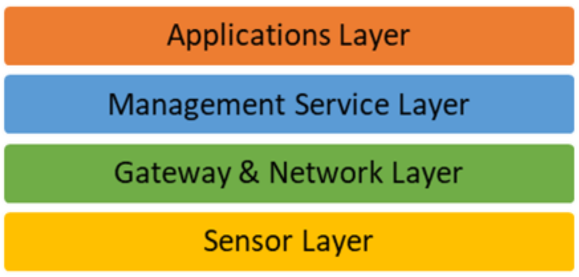


Fig. 1: General and popular IoT architecture

[*] Corresponding Author

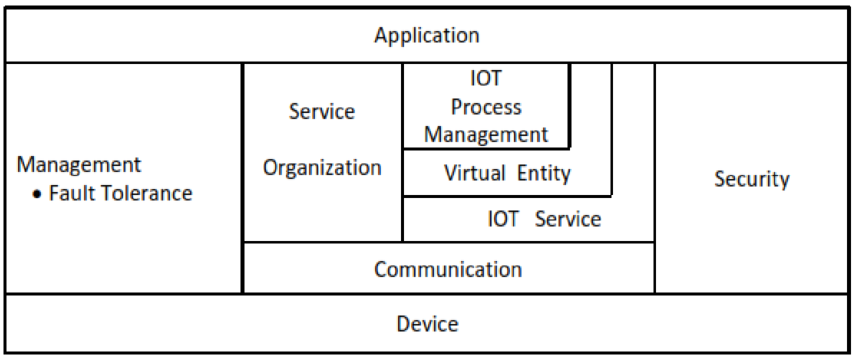


Fig. 2: IoT reference architecture

Another architecture, which is known as the reference architecture for IoT is described in [2]. This architecture can be considered as the expanded edition of the general architecture of IoT. An illustration of this is presented in Fig. 2.

Due to the existence of several IoT networks and technologies, its management is rather complex. One of the mechanisms to increase system dependability is fault tolerance (FT). Fault Tolerance is a basic and essential necessity in improving the dependability of system management. Fault tolerance management ensures that the overall performance of the system is not impaired by faults and the system is able to function normally, hence defects are not transferred to the final output.

Several phases have been proposed to increase the fault tolerance of a system: fault prediction; fault prevention; fault detection; fault isolation; and fault recovery. When fault occurs in a complicated system the importance of fault tolerance is more evident. If faults are not corrected and transferred to the output, it will cause the whole system to malfunction. So the design and implementation should be such that the emergence and fault possibility is predicted and prevented from occurring. And if a fault arises, the system must be able to promptly detect, isolate, and then recover it in the next step.

In each of the phases, various techniques and strategies are applied. The origin of faults should be determined in the first layer of the IoT architecture because if such are not addressed in the sensing layer, they will likely spread to higher layers of the propagation path. Thus the faults should be predicted, detected and resolved at the time of emergence.

Moreover, fault tolerance is a qualitative parametric system and cannot be considered definitively. This parameter is considered a crucial factor in modern systems that their complexity and intelligence is increasing. Fuzzy logic should be used to evaluate the qualitative parameters that exist in the real world. Linguistic parameters are different from numerical parameters and for measuring such qualitative factors the fuzzy logic should be used. In fact, fuzzy logic is very flexible and has the ability to deal with inaccurate data and can easily be integrated with control systems and used to manage the system. This logic is a proper tool to work under conditions of uncertainty and non-linear conditions and has the ability to model the qualitative parameters.

In this paper, it is assumed that the Wireless Sensor Networks (WSN) is used in the sensing layer of IoT architecture according to the levels of fault propagation from lower to higher layers. The fault in these networks should be managed and controlled in the first layer IoT architecture. So far, various architectures and protocols have been proposed to manage the fault in WSN. A management system that can evaluate or estimate the increase in the fault tolerance of each of these architectures is the main objective of this study. The designed FT evaluator should first act based on fuzzy logic, then it should examine the increase in FT of each of the proposed architectures and protocols in this area based on different techniques and procedures.

The rest of this article analyzes the IoT architecture and fault tolerance management position in the architecture and describes the fault-tolerant protocols and management frameworks offered for wireless sensor networks. Then a variety of fuzzy methods to increase FT ability are explained. In latter stages, the fuzzy comparison and evaluation of the protocols and frameworks increasing FT feature are presented, the proposed FISWSN framework is fully defined and explained, and the fuzzy comparison and evaluation of the protocols and frameworks increasing FT feature are presented. The last part of this paper presents the conclusion and strategies for future works in this area.

II. RELATED WORKS

A. *Fault tolerance in the IoT Architectures*

The most important factor in IoT evaluation is its architecture. And the main aspect to consider is the components used in the different layers. IoT components considered by the author in [3] include RFID, WSN, Addressing Share, Data Storage and Virtualization. In [1] the components of IoT such as Identification, Sensing, Communication, Computation, Services and Semantics are introduced. Looking at each of the IoT components must be mutual e.g. WSN. These networks are independent of each other and have their own unique architecture.

Network nodes can communicate with each other based on defined standards. But when the wireless sensor networks are interconnected as part of an IoT network, they should be viewed from a different standpoint. Standards, architectures, and protocols will vary. Naturally, these changes will also influence strategies and FT enhancement techniques.

The first phase in increasing the FT feature of a wireless network system is fault prediction and a smart method of which is described in [4]. This particular technique brings about constant and real-time system monitoring. A 4-layer architecture design described in [4] is used to predict fault by monitoring conditions in the data transmission with comprehensive and reliable data processing. These types of architectures are usually impractical in many systems including Real Time systems because they are costly and increase system response time. In addition, if fault occurs for any reason despite previous predictions, some measures should be taken so that fault is discovered and restored. The proposed architecture in [4] is purely focused on the fault prediction phase.

Fault tolerance techniques for Internet of Military Things (IOMT) are discussed in [5]. In this study, some special features of FT techniques are considered with a holistic approach. In [5] the fault models in the sensing layer of IOMT architecture are divided into three categories. The first group consists of faults that threaten processors or microcontrollers. The second group is about faults that may occur in sensors, and the third group focuses on the simultaneous occurrence of faults in microcontrollers and sensors. In the second layer of IOMT architecture, the fault occurrence in the communication nodes and links is considered. In this architecture MM approach is used for fault detection. This approach is presented by Malek and Maeng. In MM, the same input is sent to a pair of processor and the output of both processors is compared for changes that could indicate fault. The method discussed in the recovery phase is the removal of faulty nodes. Naturally, this method has its own disadvantages because the fault is just removed and not resolved. System reconfiguration transpires in the final step.

Fault-tolerant routing is an approach described in [6]. Authors in this paper have considered the delivery of data packets between a pair of source and destination even in the presence of faults. The designed algorithm in [6] has acted based on the Learning Automata and has a special value. The algorithm is able to deliver the packets with a high degree of reliability in a heterogeneous environment. The results of the implementation of this method indicate a reduced overhead on the system, which reduces network energy consumption.

Layer based fault tolerance management is the design defined for the end to end transmission in [7]. In this design the author is focused on the detection phases and fault location. In this paper, fuzzy logic is used to locate the roots of the faults in the network and distinguish the real faults and false alarms. After fault detection and location, the fault recovery is done. It is worth noting that in this structure an FCM-Base monitoring model is applied.

The use of virtual services concept is introduced in [8]. In this method, the virtual services are used to replace the faulty nodes. In virtual services, the data of two or more sensors is used. Also, in order to find and elect the virtual or actual optimized sensor, the genetic algorithm called NSGA-ii, is used.

Decentralized fault tolerance mechanism is introduced in the form of an intelligent middleware for IoT in [9]. In this approach, redundancy is defined at the service level. In fact, the services are considered as two versions. A corresponding service is maintained for each version so that in the event of a system fault, recovery and restoration automatically occurs within a few seconds.

A new Fault-tolerance architecture is introduced for IoT in [10]. WSN is used in the first layer of the architecture which is used in monitoring condition. Fault tolerance in the architecture is through redundancy in the communication paths. Backup communication paths are utilized in case of traffic bottlenecks or other complications that threaten paths between nodes.

B. *Fault tolerance in the first layer*

Fault tolerance mechanisms in wireless sensor networks as described in [11] are divided into three main classes: Redundancy Based; Clustering Based; and Deployment Based. Specific techniques are used in each of these groups. For example, in cluster based wireless sensor networks the recovery techniques are different compared to the other classes. In these networks, nodes are divided into two types: Cluster Head; and Non-Cluster Head. Cluster Head nodes are abbreviated as CH and the Non-Cluster Head nodes are abbreviated as NCH. Cluster Head nodes are more important than Non-Cluster Head nodes because they contain the latest aggregated data.

Network nodes detect fault in their hardware or software components, this Self-Detection technique is introduced in [12]. If a neighboring node plays a role in fault detection, this method is called Group Detection, and is fully described in [13, 14]. The third method used for fault detection is Hierarchical Detection, which is applicable for hierarchical clustered networks as mentioned in [15, 16].

One NCH nodes recovery technique is the method of ignoring the output generated by the node and is introduced in [12]. Henceforth, this method is briefly referred to as Ignore. Another technique for recovery of the NCH nodes is introduced in [17, 18]. Using "majority vote" system, the data generated by the faulty node is removed in the final output and aggregated data. A third method of recovery of the NCH nodes, which can be described as communication redundancy, is explained in [19]. The source can use more than one communication path to the destination nodes. If the main path is faulty for any reason, or if there is congestion or one of the intermediate nodes is lost, the node could transfer data through the alternative path.

In CH nodes recovery, the first step is the election of the alternative node. If the faulty CH elects the alternative node itself, this method is called Self-Election. This is studied in [20]. Members of the same node could also elect replacement for the faulty CH node. This is called Group – Election and is introduced in [21]. A third alternative method presented in [22, 23] is where the alternative node is elected by the central manager hierarchically, and is called Hierarchical Election.

The second phase is the recovery of CH nodes which starts after the election alternative CH node. PreCopy technique, introduced in [24] is done when the latest version of the aggregated data is copied into the new candidate cluster node. The new cluster head is responsible in sending the aggregated data to the sink. Another method discussed by the authors in [25] is Code Distribution, where tasks, operations, and aggregated data from NCHs are directed to the new CH. The last approach presented in this field is called Remote Execution, defined in [26, 27]. In this method the faulty CH node is controlled and managed remotely by another alternative node.

C. *Fault Tolerance Protocols*

FT-DSC protocol, introduced in [28] has created FT capability for both NCH and CH nodes. In the fault detection phase the techniques Self-Detection and Hierarchical-Detection techniques are used simultaneously. In the fault recovery phase, Ignore Value from faulty Node Technique is used for the NCH nodes, while CH nodes are recovered by electing the alternative node through Hierarchical Election method, and then using Remote Execution method for service distribution and implementation on new CH.

FTPASC is a protocol that has created FT feature for CH nodes. This protocol, introduced in [29] elects an “organizer node” for each cluster, also known as the redundant node. If for any reason the cluster head fails, the organizer node quickly replaces this node and undertakes its tasks. This process utilizes the Group Detection method, and code distribution is implemented in the service distribution at the recovery phase.

The importance of FT feature in CH nodes is imperative because they are responsible for data aggregation of all NCHs in a cluster. CMATO protocol, introduced in [30] has considered this important point significantly. In this protocol, network nodes are responsible to detect their neighbor CHs. In the event of a CH node failure, NCH nodes of that cluster could subscribe to one of the neighbor nodes and transmit their data to its CH. Thus, the fault detection is done by Group Detection, the fault recovery by Group Election, and the Code Distribution is applied in the Service Distribution stage.

In the protocol introduced in [31], a new approach called Active Caching is used. In this protocol, the aggregated data is stored in the middle nodes between the source and the sink so that it can be reused, circumventing the need to repeat data aggregation operations. This protocol has applied Hierarchical Detection technique in the fault detection phase. After electing the new cluster head using Hierarchical Election, Service distribution is done by Precopy technique.

The techniques used in the fault detection and recovery phases are the same for both LRC and FT-DSC protocols. The difference between these two protocols is in their network re-clustering. LRC protocol is purely implemented locally, while in, FT_DSC protocol, it is done simultaneously through dynamic and static methods.

Fault tolerance management frameworks introduced in [18, 34-36] have provided FT increase at different levels of the network. Ridesharing has created FT feature at communications level of the network but GHT and Craft have created this feature at network node level. Marzulla, presented in [18] has not created FT in Node or Network layers completely. Craft, introduced in [35] has provided fault detection feature in combination with Self-Detection and Hierarchical-Detection techniques. But Marzulla, GHT and Ridesharing frameworks have just used one of the fault detection techniques. GHT has used Hierarchical - Detection technique and Ridesharing has used Self-Detection technique.

In the fault recovery phase, Ridesharing framework has used a combination of Precopy and Remote Execution techniques. However, GHT and Craft have implemented fault recovery just for CH nodes and Marzulla has only provided this feature at NCH nodes. The framework presented in [36] known as ECraft, is relatively more complete and stronger than the earlier proposed frameworks. ECraft has created FT feature both at network nodes and network communications levels. In the fault detection phase it has used a combination of all three mentioned techniques. It has also used Precopy and Remote Execution techniques simultaneously.

The proposed protocol in [37] has features to increase the FT capability of a WSN and provides the foundation to increase the lifespan of the network. The proposed algorithm is known as FFT-Leach. The NCH groups use fuzzy algorithm to elect and join the best CH based on distance and energy.

D. *Optimal Choice method*

The algorithm proposed in [38] has increased fault tolerance in WSN using fuzzy logic. In this algorithm the area covered by the sensor that has been damaged is covered by the neighboring nodes. The election and replacement of the neighboring node is done by fuzzy logic and based on a series of parameters. These parameters include the distance between the faulty node and the neighbor node, distance from the cluster head node, remaining energy and the overlapping range of the sensor.

Another fuzzy method is proposed to detect faulty nodes in the clustered wireless sensor networks in [39]. In the proposed method, the value sensed by other groups and the rate of fault in the nodes provide input to the Fuzzy Inference System.

In the method proposed in [40], the lost area covered by the faulty node is covered by neighboring nodes. The least number of neighboring nodes is used to cover the area. After a certain node becomes faulty, the main concern is choosing the neighboring nodes to cover the affected area. The first priority in selection is the distance between the neighboring and the faulty nodes. In [40], fuzzy logic and mathematical methods are used to determine the alternative node.

Knowledge Based control system, known as FKBC is presented in [41]. The system provides an intelligent fault-tolerant version capable of detecting the Byzantine-type faults in clustered wireless sensor networks. The proposed control approach has been based on fuzzy logic and has detection capability of faulty communications between wireless sensor network nodes.

The proposed Fault management design has a Fuzzy Inference system based on fuzzy logic presented in [42]. Sensor, battery, and engine input receiver conditions have been fuzzy inferenced. Each of the nodes could have four operating modes. The network node could play its role in normal mode or act as an end node. It might be a dead node and finally the node may act as a traffic node.

Fuzzy logic is always one of the best methods to find an optimal approach. This is clearly shown in [43-45]. Finding the optimal path and using different search algorithms have been considered by the authors. If the number of nodes is increased, the comparisons and search time will have a sharp increase; thus to find the best path fuzzy modeling is used. Optimal fuzzy decision-making has been the purpose of this article [44]. The authors have shown that by using tolerance approach and fuzzy set theory better results could be obtained. Finding the minimum and optimum cost is considered in [45]. The proposed model involves a valuable linear performance for fuzzy equations.

III. Proposed Method

Fault tolerance capability is one of the dependability enhancement mechanisms of each system and should be integrated in different layers of the IoT architecture. This paper is focused on the first layer, i.e. the sensing layer. The purpose of the proposed framework is to evaluate the FT capability of any framework or protocol. At different applications, sensors, and environments in which wireless sensor networks are used, FT feature measurement enables the system's capability to reach its maximum performance while cost is minimized by the correct and optimal choice of protocols. In addition, by increasing FT capability in the first layer, system dependability and efficiency is increased.

The proposed framework, called FISWSN is designed in a specific way to measure the fault tolerance created by each of the architectures and protocols in the wireless sensor networks. Each of these architectures and protocols has its own unique attributes and their FT is measurable. Due to the fact that an FT is a qualitative parameter, its measurement can be done by fuzzy logic. In the proposed FISWSN design, the system can make decisions about its FT capability based on the techniques used by each architecture or protocol in fault detection and recovery phase. Enabling FT in the propagation level will be effective as a third index. This may already be happening at the network nodes. Generally, the number and type of employed techniques are among the major indicators in this assessment. These techniques can be implemented individually or in combination.

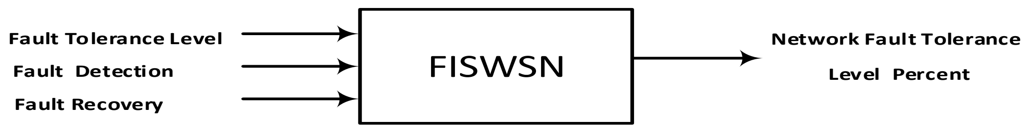


Fig. 3: The overall view of the proposed FISWSN framework

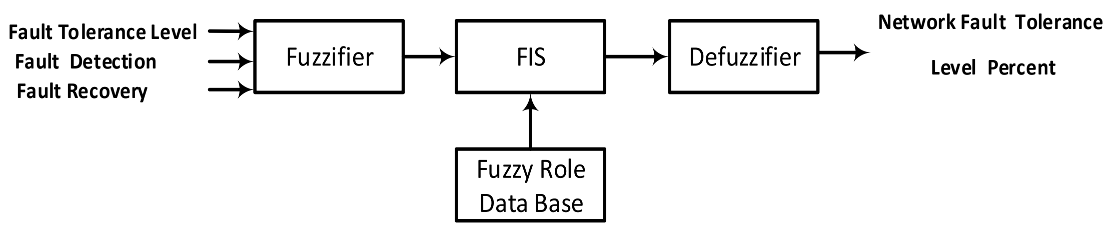


Fig. 4: Components of FISWSN framework

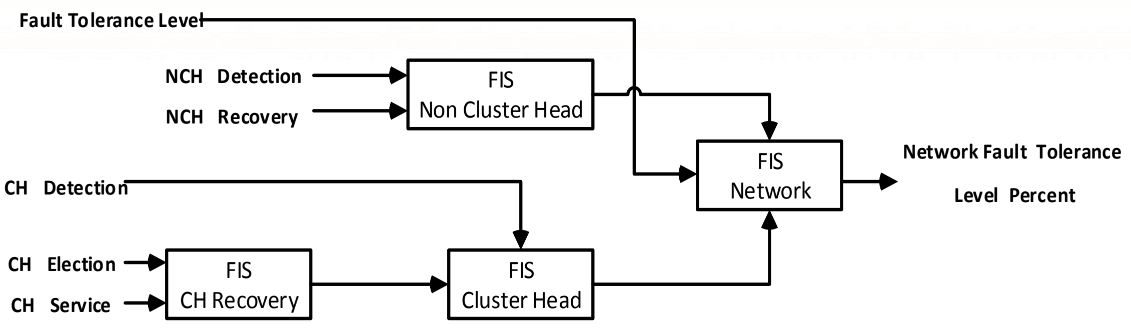
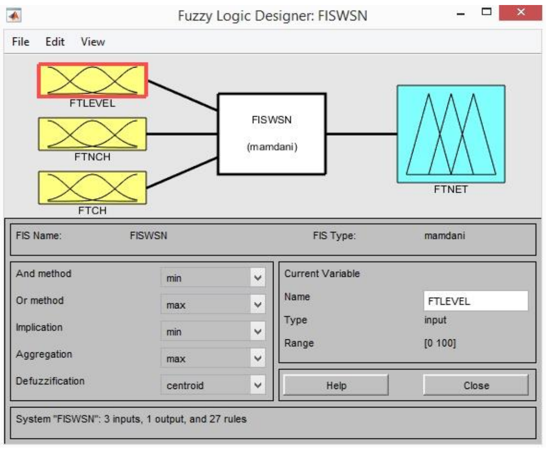


Fig. 5: Subsystems forming FISWSN

Fig. 3 presents the view of the proposed FISWSN framework. Various components constituting the FISWSN are presented in Fig. 4. It should be noted that FISWSN itself consists of a number of fuzzy subsystems. The subsystems forming FISWSN are presented in Fig. 5.

IV. SIMULATION

Designing a fuzzy FISWSN system in MATLAB is presented in Fig. 6. The role of fuzzifier is converting the input variables into linguistic variables. It should be noted that this operation is performed by defined membership functions. The output is delivered to the inference engine and using the rules stored in the database, the engine generates the desired output. The main engine database rules are listed in Table 1.


Fuzzy Logic Designer: FISWSN

FIS Name: FISWSN   FIS Type: mamdani

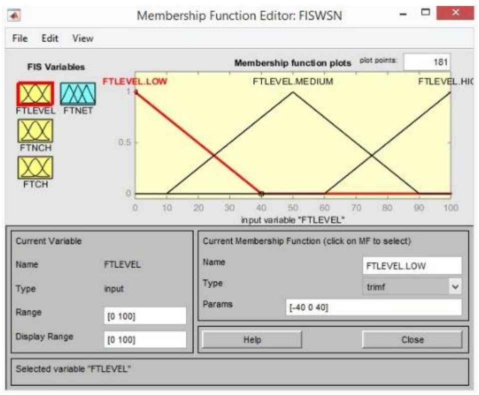
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: FTLEVEL
Type: input
Range: [0 100]

System "FISWSN": 3 inputs, 1 output, and 27 rules

Fig. 6: FISWSN system implementation in MATLAB

The output generated by FISWSN fuzzy inference engine is delivered to the related defuzzifier. The role of the defuzzifier is to convert the fuzzy engine output to a Crisp value. Finally, this is transferred to the system as the final output through the defuzzifier. Figures (7-10) present the membership functions of the system inputs and outputs in MATLAB environment. Figures (11-13) have presented the implementation of FISWSN subsystems in this environment.

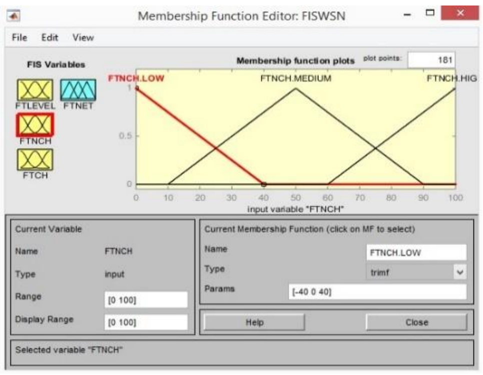

Membership Function Editor: FISWSN

Current Variable
Name: FTLEVEL
Type: input
Range: [0 100]
Display Range: [0 100]

Current Membership Function (click on MF to select)
Name: FTLEVEL.LOW
Type: trimf
Params: [-40 0 40]

Selected variable "FTLEVEL"

Fig. 7: Implementation of the membership functions of the inference engine first input (FT-Level)

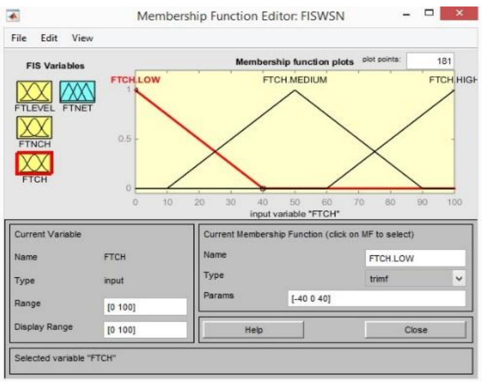

Membership Function Editor: FISWSN

Current Variable
Name: FTNCH
Type: input
Range: [0 100]
Display Range: [0 100]

Current Membership Function (click on MF to select)
Name: FTNCH.LOW
Type: trimf
Params: [-40 0 40]

Selected variable "FTNCH"

Fig. 8: Implementation of the membership functions of the inference engine second input (FT-Detection)


Membership Function Editor: FISWSN

Current Variable
Name: FTCH
Type: input
Range: [0 100]
Display Range: [0 100]

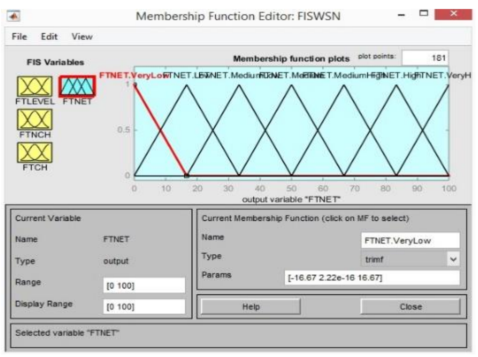
Current Membership Function (click on MF to select)
Name: FTCH.LOW
Type: trimf
Params: [-40 0 40]

Selected variable "FTCH"

Fig. 9: Implementation of the membership functions of the inference engine third input (FT-Recovery)

Table 1: FISWSN engine database rules

Fault Tolerance of WSN Fuzzy Inference System Rules				
Rule No.	**F.T Level**	**Cluster Head F.T**	**Non-Cluster Head F.T**	**Fault Tolerance Level Decision**
1	Low	Low	Low	Very Low
2	Low	Low	Medium	Low
3	Low	Low	High	Low Medium
4	Low	Medium	Low	Low
5	Low	Medium	Medium	Low Medium
6	Low	Medium	High	Medium
7	Low	High	Low	Low Medium
8	Low	High	Medium	Medium
9	Low	High	High	High Medium
10	Medium	Low	Low	Low
11	Medium	Low	Medium	Low Medium
12	Medium	Low	High	Medium
13	Medium	Medium	Low	Low Medium
14	Medium	Medium	Medium	Medium
15	Medium	Medium	High	High Medium
16	Medium	High	Low	Medium
17	Medium	High	Medium	High Medium
18	Medium	High	High	High
19	High	Low	Low	Low Medium
20	High	Low	Medium	Medium
21	High	Low	High	High Medium
22	High	Medium	Low	Medium
23	High	Medium	Medium	High Medium
24	High	Medium	High	High
25	High	High	Low	High Medium
26	High	High	Medium	High
27	High	High	High	Very High



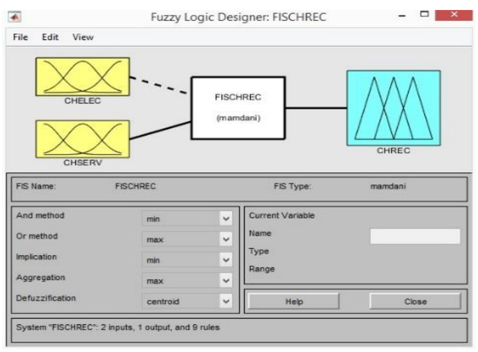
Current Variable
Name FTNET
Type output
Range [0 100]
Display Range [0 100]

Current Membership Function (click on MF to select)
Name FTNET.VeryLow
Type trimf
Params [-16.67 2.22e-16 16.67]

plot points: 181

Selected variable "FTNET"

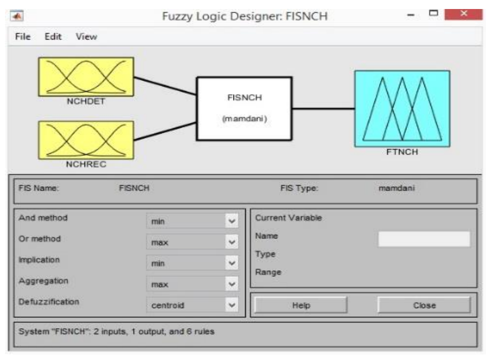
Fig. 10: Implementation of the membership functions of the inference engine output (Network Fault Tolerance)



FIS Name: FISCHREC FIS Type: mamdani
And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid

System "FISCHREC": 2 inputs, 1 output, and 9 rules

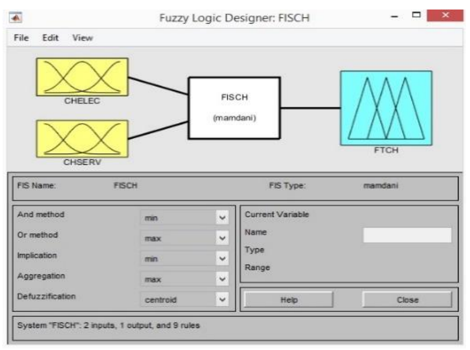
Fig. 11: Implementation of FIS-CH-Recovery engine in MATLAB environment



FIS Name: FISNCH FIS Type: mamdani
And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid

System "FISNCH": 2 inputs, 1 output, and 6 rules

Fig. 12: Implementation of FIS-NCH engine in MATLAB environment



FIS Name: FISCH FIS Type: mamdani
And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid

System "FISCH": 2 inputs, 1 output, and 9 rules

Fig. 13: Implementation of FIS-CH engine in MATLAB environment

Database rules of the engines in subsystems FIS-CH-Recovery, FIS-NCH and FIS-CH are listed in tables 2, 3 and 4 respectively.

Table 2: Database rules of FT-CH-Recovery engine

CH - Recovery		
Election	**Service**	**Decision**
Low	Low	Low
Low	Medium	Low
Low	High	Low
Medium	Low	Low
Medium	Medium	Medium
Medium	High	Medium
High	Low	Low
High	Medium	Medium
High	High	High

Table 3: Database rules of FT-NCH engine

F.T - NCH		
Detect	**Recovery**	**Decision**
Low	Low	Low
Low	High	Low
Medium	Low	Low
Medium	High	Medium
High	Low	Low
High	High	High

Table 4: Database rules of FT-CH engine

F.T - CH		
Detect	**Recovery**	**Decision**
Low	Low	Low
Low	Medium	Low
Low	High	Low
Medium	Low	Low
Medium	Medium	Medium
Medium	High	Medium
High	Low	Low
High	Medium	Medium
High	High	High

Table 5: FT feature describer

Status	Range	Percent
Very Low	0 - 10	5%
Low	10 - 20	15%
Medium Low	20 - 40	30%
Medium	40 – 60	50%
Medium High	60 – 80	70%
High	80 – 90	85%
Very High	90 - 100	95%

Based on the defuzzifier output, the fault tolerance of a system could be evaluated using Table 5. If the output value is from 0 to10, the FT capability of the system can be classified as very low, while an output in the range of 40-60 is described as medium.

The inputs of the proposed FISWSN framework are converted into linguistic variables using Triangle membership functions. In contrast, the FISWSN engine outputs are converted into a Crisp value by Trapezoid membership functions. In trapezoidal membership functions, a, b, c and d are used to represent the boundary conditions; while in triangular functions the boundary conditions a, b and c are used. Input and output variables use the following tags:

- **Input**

Fault Tolerance Level = {Low, Medium, High} = {LP, MP, HP}

Fault Detection = {Low, Medium, High} = {LD, MD, HD}

Fault Recovery = {Low, Medium, High} = {LR, MR, HR}

- **Output**

Network Fault Tolerance = {Very Low, Low, Low Medium, Medium, High Medium, High, Very High} = {VL, L, LM, M, HM, H, VH}

FISWSN inputs are defined as "Low", "Medium" and "High" by linguistic term sets. This membership function is presented as follows.

First Input Membership Functions:

$$\mu_{FT\_L\_H} = \begin{cases} 0 & ; \quad x \leq a \\ \frac{x-a}{b-a} & ; \quad a \leq x \leq b \\ 1 & ; \quad x > b \end{cases}$$

$$\mu_{FT\_L\_M} = \begin{cases} 0 & ; \quad x < a \\ \frac{x-a}{b-a} & ; \quad a \leq x < b \\ 1 & ; \; b \leq x < c \\ \frac{d-x}{d-c} & ; \; c \leq x \leq d \\ 0 & ; \quad x > d \end{cases}$$

$$\mu_{FT\_L\_L} = \begin{cases} 1 & ; \quad x < a \\ \frac{b-x}{b-a} & ; \quad a \leq x \leq b \\ 0 & ; \quad x > b \end{cases}$$

Second Input Membership Functions:

$$\mu_{FT\_D\_H} = \begin{cases} 0 & ; \quad x \leq a \\ \frac{x-a}{b-a} & ; \quad a \leq x \leq b \\ 1 & ; \quad x > b \end{cases}$$

$$\mu_{FT\_D\_M} = \begin{cases} 0 & ; \quad x < a \\ \frac{x-a}{b-a} & ; \quad a \leq x < b \\ 1 & ; \; b \leq x < c \\ \frac{d-x}{d-c} & ; \; c \leq x \leq d \\ 0 & ; \quad x > d \end{cases}$$

$$\mu_{FT\_D\_L} = \begin{cases} 1 & ; \quad x < a \\ \frac{b-x}{b-a} & ; \quad a \leq x \leq b \\ 0 & ; \quad x > b \end{cases}$$

Third Input Membership Functions:

$$\mu_{FT\_R\_H} = \begin{cases} 0 & ; \quad x \le a \\ \frac{x-a}{b-a} & ; \quad a \le x \le b \\ 1 & ; \quad x > b \end{cases}$$

$$\mu_{FT\_R\_M} = \begin{cases} 0 & ; \quad x < a \\ \frac{x-a}{b-a} & ; \quad a \le x < b \\ 1 & ; \quad b \le x < c \\ \frac{d-x}{d-c} & ; \quad c \le x \le d \\ 0 & ; \quad x > d \end{cases}$$

$$\mu_{FT\_R\_L} = \begin{cases} 1 & ; \quad x < a \\ \frac{b-x}{b-a} & ; \quad a \le x \le b \\ 0 & ; \quad x > b \end{cases}$$

Table 5 is used to classify the FT feature of networks based on FISWSN output. Output membership functions are defined as follows.

$$\mu_{FT\_VH} = \begin{cases} 0 & ; \quad x \le a \\ \frac{x-a}{b-a} & ; \quad a \le x \le b \\ 1 & ; \quad x > b \end{cases}$$

$$\mu_{FT\_H} = \begin{cases} 0 & ; \quad x < a \\ \frac{x-a}{b-a} & ; \quad a \le x < b \\ \frac{d-x}{d-c} & ; \quad c \le x \le d \\ 0 & ; \quad x > c \end{cases}$$

$$\mu_{FT\_HM} = \begin{cases} 0 & ; \quad x < a \\ \frac{x-a}{b-a} & ; \quad a \le x < b \\ \frac{d-x}{d-c} & ; \quad c \le x \le d \\ 0 & ; \quad x > c \end{cases}$$

$$\mu_{FT\_M} = \begin{cases} 0 & ; \quad x < a \\ \frac{x-a}{b-a} & ; \quad a \le x < b \\ \frac{d-x}{d-c} & ; \quad c \le x \le d \\ 0 & ; \quad x > c \end{cases}$$

$$\mu_{FT\_LM} = \begin{cases} 0 & ; \quad x < a \\ \frac{x-a}{b-a} & ; \quad a \le x < b \\ \frac{d-x}{d-c} & ; \quad c \le x \le d \\ 0 & ; \quad x > c \end{cases}$$

$$\mu_{FT\_L} = \begin{cases} 0 & ; \quad x < a \\ \frac{x-a}{b-a} & ; \quad a \le x < b \\ \frac{d-x}{d-c} & ; \quad c \le x \le d \\ 0 & ; \quad x > c \end{cases}$$

$$\mu_{FT\_VL} = \begin{cases} 1 & ; \quad x < a \\ \frac{b-x}{b-a} & ; \quad a \le x \le b \\ 0 & ; \quad x > b \end{cases}$$

In FISWSN engine design, the composition based inference engine is applied. Also the applied rule is Modus Ponens presented in equation (1).

$$\mu_{B'}(y) = \mathrm{Sup}\ t[\mu_{A'}(x), \mu_{A\to B}(x,y)] \qquad \text{Equation (1)}$$

The applied implication is Mamdani Minimum.

$$\mu_{Q_{MM}}(x,y) = \min[\mu_{FP_1}(x), \mu_{FP_2}(y)] \qquad \text{Equation (2)}$$

Where $FP_1(x)$, Fuzzy Proposition is the premise and $FP_2(y)$ Fuzzy Proposition is conclusion. For all the T-norms of the min operator and for all S-norms the Max is used. FISWSN engine fuzzifier is considered as Single Tone fuzzifier.

$$\mu_{A'}(X) = \begin{cases} 1\,; & X = X^* \\ 0\,; & Otherwise \end{cases} \qquad \text{Equation (3)}$$

Finally, the proposed engine defuzzification is the average defuzzification of the centers.

$$y^* = \frac{\sum_{l=1}^{M} \bar{y}^l w_l}{\sum_{l=1}^{M} w_l} \qquad \text{Equation (4)}$$

Where y is the fuzzy set I and W is height of fuzzy set L. For FISWSN engine design the equations 1 and 2 are combined to obtain equation (5).

$$\mu_{B'}(y) = \mathrm{Sup}\ t[\mu_{A'}(x), \min[\mu_A(x), \mu_B(y)] \qquad \text{Equation (5)}$$

If the rule composition happens in equation (5) and min operator is used for t-norms, the equation (6) is obtained.

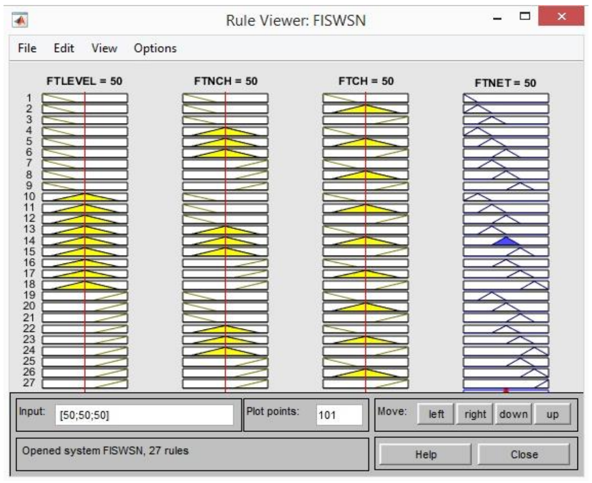
$$\mu_{B'}(y) = \max_{1\le l\le M} [\mathrm{Sup}\ \min[\mu_{A'}(x), \mu_{A_1^l}(x_1),. \ ,\mu_{A_n^l}(x_n)\mu_{B^l}(y)]] \qquad \text{Equation (6)}$$

If the fuzzy set A is a single tone set (equation (3)), then the equation (6) is converted into equation (7) that this equation is called minimal inference.

$$\mu_{B'}(y) = \max_{1\le l\le M} [\min[\mu_{A_1^l}(x_1^*),\ldots\ldots\ldots,\mu_{A_n^l}(x_n^*)\mu_{B^l}(y)]] \qquad \text{Equation (7)}$$

If the Crisp value of the system output is to be found by the mean defuzzifier i.e. by equation (4), the overall output of the FISWSN engine can be obtained by equation (8).

$$y^* = \frac{\sum_{l=1}^{M} \bar{y}^l (\min_{1\le i\le n} \mu_{A_i^l}(x_i))}{\sum_{l=1}^{M} (\min_{1\le i\le n} \mu_{A_i^l}(x_i))} \qquad \text{Equation (8)}$$



Input: [50;50;50]
Plot points: 101
Move:
Opened system FISWSN, 27 rules

Fig. 14: FISWSN engine rules

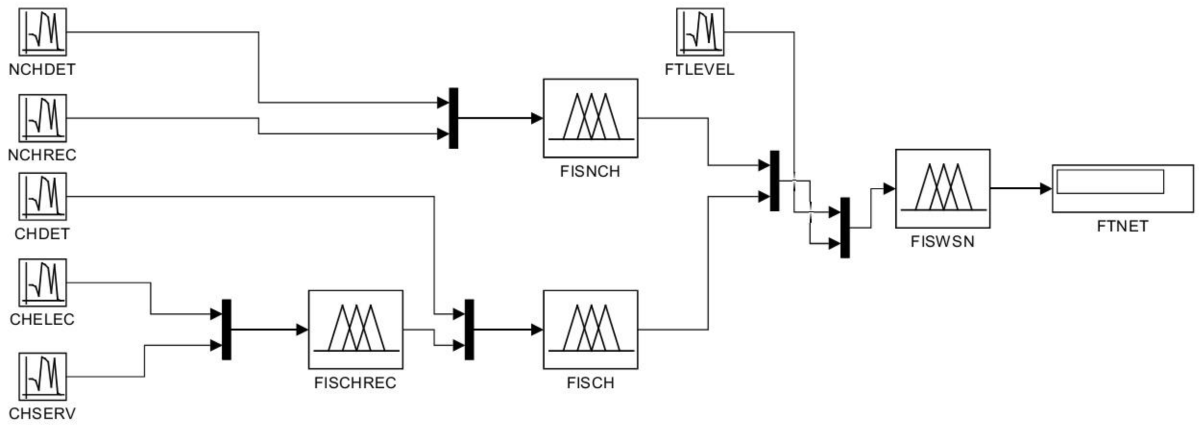


Fig. 15: FISWSN engine implementation in Simulink environment

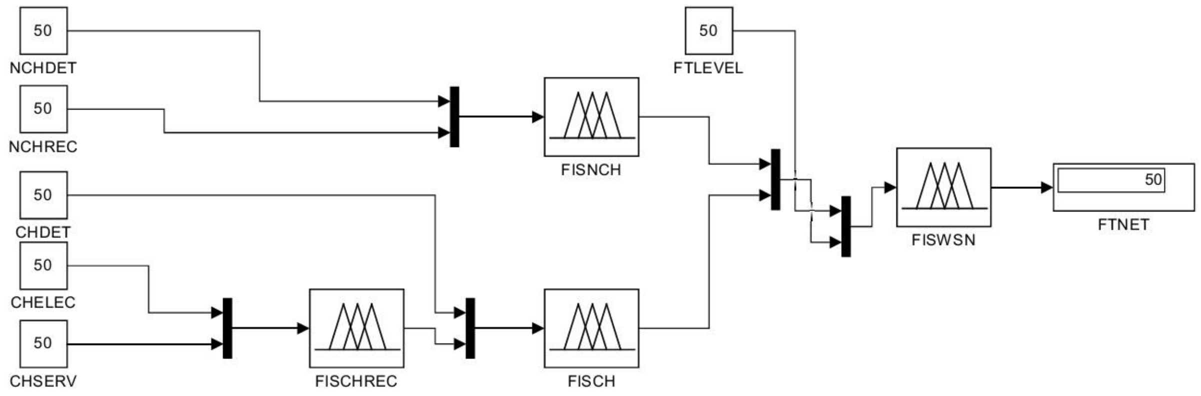
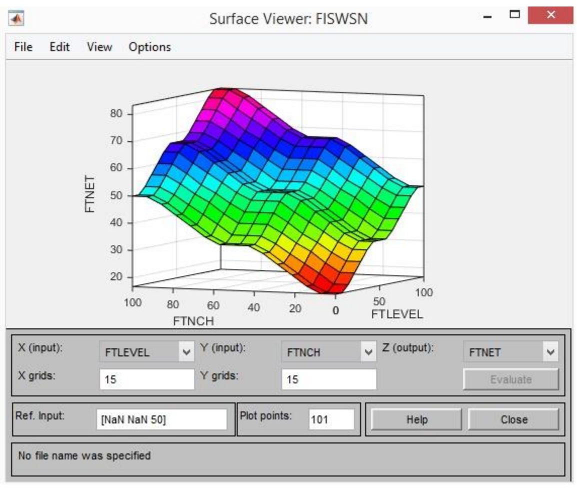


Fig. 16: FISWSN engine calibration


Surface Viewer: FISWSN

X (input): FTLEVEL
Y (input): FTNCH
Z (output): FTNET
X grids: 15
Y grids: 15
Ref. Input: [NaN NaN 50]
Plot points: 101

No file name was specified

Fig. 17: Surface View of FISWSN engine

Fig. 14 presents a view of the database rules of the inference engine FISWSN implemented in MATLAB. While Fig. 17 presents a surface view of the FISWSN Fuzzy Systems.

Fig. 15 is an illustration of FISWSN system implementation in Simulink environment. Also in Fig. 16, a view of the engine calibration is presented.

V. DISCUSSION

The proposed FISWSN framework is of vital importance. Since faults originate from lower levels and propagate to higher ones, this fact has been put into consideration and as such, FISWSN has adopted FT management in the first layer of IoT architecture. Likewise, in accordance to the techniques referred to in [12-16], the same have been proposed in [17-19] for the recovery of the NCH nodes, and similarly for the recovery of the CH nodes proposed in [20-27], the FT capability is established in the WSN clustering networks. Thus, the proposed FISWSN framework has developed a system where the protocols are assessed, and their FT capability measured by the architecture by itself.

Fault tolerance protocols presented in [28-32], and the frameworks presented in [18, 33-37] have increased FT feature in WSN and in the first layer of the IoT architecture that it becomes significant to be able to measure this feature. Using the FISWSN framework, it is possible to measure the increase in FT capability in these types of networks. These comments and assessments are based on the number and types of techniques applied in each of them. Table 6 presents the evaluation and assessment based on the fuzzy logic of the protocol and the different fault tolerance frameworks of the wireless sensor networks.

Table 6: Fuzzy assessment of different FT protocols and frameworks

	Protocol / Framework	**Fault Tolerance Level**	**Fault Detection Technique**	**Fault Recovery Technique**	**Network Fault Tolerance Percent**
Fault Tolerance Protocols	FT – DSC [28]	Node CH & NCH	Self & Hierarchical	Ignore Value & Remote exe	38.04
	FTPASC [29]	Node CH only	Group	Code Distribution	33.21
	CMATO [30]	Node CH only	Group	Code Distribution	33.21
	Active Caching [31]	Node CH only	Hierarchical	PreCopy	32.23
	LRC [32]	Node CH & NCH	Self & Hierarchical	Ignore Value & Remote exe	38.04
Fault Tolerance Frameworks	Marzullo [18]	Incomplete	Hierarchical	Aggregation	19.95
	RideSharing [33]	Communication	Self	PreCopy & Remote exe	30.73
	GHT [34]	Node	Hierarchical	PreCopy	35.22
	Craft [35]	Node	Self & Hierarchical	PreCopy	36.24
	ECraft [36]	Node & Communication	Self & Group & Hierarchical	PreCopy & Remote exe	50.99

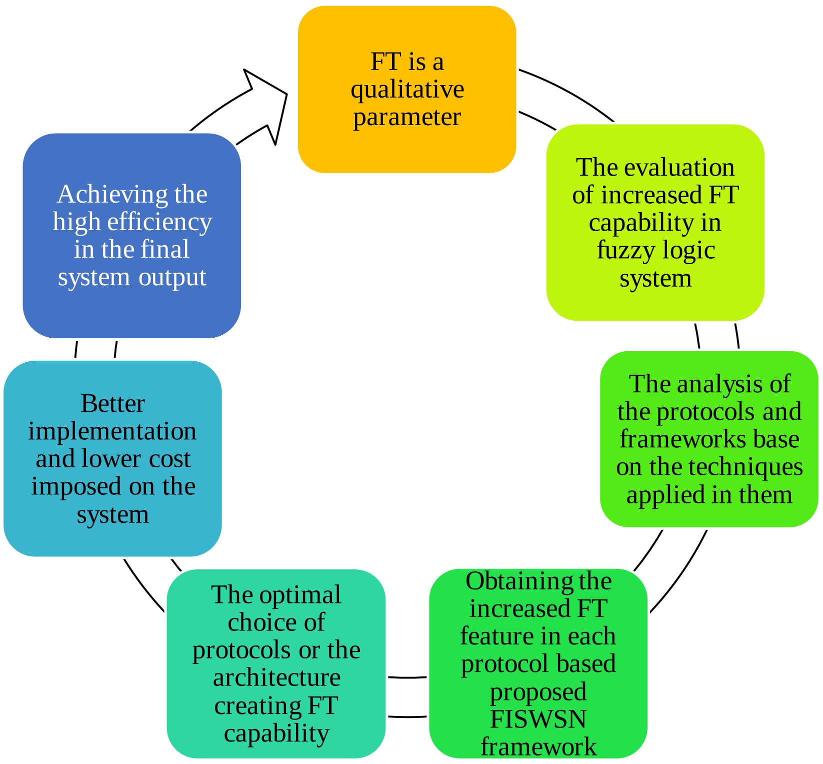

FT is a qualitative parameter
The evaluation of increased FT capability in fuzzy logic system
The analysis of the protocols and frameworks base on the techniques applied in them
Obtaining the increased FT feature in each protocol based proposed FISWSN framework
The optimal choice of protocols or the architecture creating FT capability
Better implementation and lower cost imposed on the system
Achieving the high efficiency in the final system output

Fig. 17: FISWSN workflow

The next step after evaluation and assessment of the features of each protocol is to choose a protocol and architecture for the network. As this choice is more optimal, its implementation will be more affordable, and most importantly the total efficiency will be significantly increased. In [43-45], the main objective is finding the optimal solution. If FISWSN is applied, it is possible to present a correct assessment of the protocol. And if good measurement of protocols is achieved, the optimum choice can be made and consequently, the system's efficiency will improve correspondingly. This workflow is presented in Fig. 17.

The focus of the proposed method is increasing IoT fault tolerance as noted in [4] which is in the Fault Prediction phase. In [5], the faults of the second layer which include the network communications have been considered in addition to the faults of the first layer of the IoT architecture. In [6, 7], emphasis on communication and redundancy in communication paths is discussed. In [8], redundancy is performed by creating virtual sensors. Then in [9], redundancy has been considered at the service level. Similar to [7 and 8], the network FT is provided by redundancy in communication paths in [10]. The key point is that FT increase has always transpired in [4-10], but there is no information on the level of increase. This could be attributed to the lack of an evaluator framework.

This FISWSN framework being discussed in IoT architecture has the capability to be extended to other architectures. For example, Gateway and Networking which is in the second IoT layer can be applied in Cloud Computing. It should be noted that all mentioned techniques in creating fault tolerance and architectures in this area are investigated in [46].

As mentioned in [11], FT enhancement mechanisms in WSN are classified in three main groups in terms of the implementation of techniques. The main limitation in this study is that the proposed framework is applicable only in the first layer of IoT network architecture and that the implemented WSN in it is cluster based. Thus, if a network is Redundancy Based or Deployment Based and has applied the techniques discussed in this paper, the proposed FISWSN cannot measure the performance of the FT feature.

VI. Conclusion

The purpose of the proposed FISWSN framework is to obtain fault-tolerance rate in each protocol and frameworks that are implemented in the sensing layer of IoT architecture. Based on the comparison conducted by means of the fuzzy logic, an optimal selection can be achieved through the implementation of protocols and architectures. If the FT capability creation protocol is selected optimally in addition to lower overhead burden on the system, higher productivity will be achieved in the final output system.

The proposed evaluator framework is only applicable to IoT networks that are clustered in the first layer of the architecture. FISWSN Framework can be developed longitudinally and transversely. It is both possible to be extended in the first layer of IoT architecture to cover other WSN networks including Redundancy Based and Deployment Based, or through the design of another framework in the second layer of IoT architecture based on techniques to improve the FT in the field of Cloud Computing.

References

[1] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M., “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications”, IEEE Communication Surveys and Tutorials, Vol. 17, no. 4, pp. 2347-2376, Nov. 2015.

[2] Bauer, M., Boussard, M., Bui, N., Loof, J., Magerkurth, C., Meissner, S., Nettstra, A., Stefa, j., Thoma, M. and Walewski, J., “IoT Reference Architecture”, Chapter 8, Enabling Things to Talk, Springer. 2013.

[3] Gubbi, J., Buyya, R., Marusic, S. and Palaniswami, M., “Internet of Things (IoT): A vision, architectural elements, and future directions”, Future Generation Computer Systems, 29, 2013, pp.1645-1660.

[4] Xu, X., Chen, T. and Minami, M., “Intelligent fault prediction system based on internet of things”, Computers and Mathematics with Applications, Elsevier 64(2012), PP.833-839.

[5] Chudzikiewicz, J., Furtak, J. and Zielinski, Z., “Fault-tolerant techniques for the Internet of Military Things”, 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 14-16 Dec. 2015. pp 496-501.

[6] Misra, S., Gupta, A., Krishna, P. V. and Agarwal, H., “An Adaptive Learning Approach for Fault-Tolerant Routing in Internet of Things”, 2012 IEEE Wireless Communications and Networking Conference (WCNC) , 1-4 April 2012, pp. 815-819.

[7] Li, X., Ji, H. and Li, Y., “Layered Fault Management Scheme for End-to-end Transmission in Internet of Things”, 2011 6th International ICST Conference on Communications and Networking in China (CHINACOM), 17-19 Aug. 2011, pp.1021 – 1025.

[8] Zhou, S., Lin, K., Na, J., Chuang, C. and Shih, C., “Supporting Service Adaptation in Fault Tolerant Internet of Things”, 2015 IEEE 8th International Conference on Service-Oriented Computing and Applications (SOCA), 19-21 Oct. 2015, pp. 65-72.

[9] H. Su, P., Shih, C., Yung-Jen Hsu, J., Lin, K., Wang, Y., “Decentralized fault tolerance mechanism for intelligent IoT/M2M middleware”, 2014 IEEE World Forum on Internet of Things (WF-IoT), 6-8 March 2014, pp.45-50.

[10] Gia, T. N., Rahmani, A., Westerlund, T., Liljeberg, P. and Tenhunen, H., “Fault tolerant and scalable IoT-based architecture for health monitoring”, 2015 IEEE Sensors Applications Symposium (SAS), 13-15 April 2015, pp. 1-6.

[11] Kakamanshadi, G., Gupta, S. and Singh, s., “A survey on fault tolerance techniques in Wireless Sensor Networks”, 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), 8-10 Oct. 2015, pp. 168 – 173.

[12] S. Harte and A. Rahman. “Fault Tolerance in Sensor Networks Using Self-Diagnosing Sensor Nodes”, In the IEEE International Workshop on Intelligent Environment, pages 7–12, June 2005.

[13] M., Ding, D., chen, K., xing and X., cheng. “Localized fault-tolerant event boundary detection in sensor networks”, in proceeding of the 24th Annual Joint Conference of the IEEE Comuter and Communications Societies (INFOCOM 105) Miami, USA, march 2005.

[14] B. Krishnamachari and S. Iyengar. “Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks”, IEEE Transactions on Computers, 53:241–250, March 2004.

[15] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. “Sympathy: a debugging system for sensor networks”, In IEEE International Conference on Local Computer Networks, 2004.

[16] S. Rost and H. Balakrishnan, “Memento: A health monitoring system for wireless sensor networks”, In SECON, 2006.

[17] D. Desovski, Y. Liu, and B. Cukic, “Linear randomized voting algorithm for fault tolerant sensor fusion and the corresponding reliability model”, In IEEE International Symposium on Systems Engineering, pages 153–162, October 2005.

[18] K. Marzullo, “Tolerating failures of continuous-valued sensors”, ACM Transactions on Computer Systems, 8(4):284–304, 1990.

[19] N. Li and J. C. Hou, “FLSS: A Fault-Tolerant Topology Control Algorithm for Wireless Networks”, In Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, pages 275–286, 2004.

[20] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy-Efficient Communication Protocol for Wireless Micro sensor Networks”, In Proceedings of the 33rd Hawaii International Conference on System Sciences, volume 8, page 8020, 2000.

[21] G. Gupta and M. Younis, “Fault-Tolerant Clustering of Wireless Sensor Networks”, Wireless Communications and Networking, 3:1579–1584, 2003.

[22] I. Gupta, D. Riordan, and S. Sampalli, “Cluster-Head Election Using Fuzzy Logic for Wireless Sensor Networks”, In Proceedings of the 3rd Annual Communication Networks and Services Research Conference, pages 255–260, 2005.

[23] J. Staddon, D. Balfanz, and G. Durfee, “Efficient Tracing of Failed nodes in Sensor Networks”, In Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, pages 122–130, 2002.

[24] C. Frank and K. Romer, “Algorithms for Generic Role Assignment in Wireless Sensor Networks”, In Proceedings of the 3rd international conference on embedded networked sensor systems, pages 230–242, 2005.

[25] P. Levis and D. Culler, “Mate: A Tiny Virtual Machine for Sensor Networks”, In ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems, pages 85–95, New York, NY, USA, 2002. ACM Press.

[26] P. Rong and M. Pedram, “Extending the lifetime of a network of battery-powered mobile devices by remote processing: A markovian decision-based approach”, In DAC ’03: Proceedings of the 40th Conference on Design automation, pages 906–911, New York, NY, USA, 2003.ACM Press.

[27] A. Rudenko, P. Reiher, G. J. Popek and G. H. Kuenning, “The Remote Processing Framework for Portable Computer Power Saving”, In SAC ’99: Processing of the 1999 ACM symposium on Applied computing, pages 365-372, New York, NY, USA, 1999. ACM Press.

[28] L., Karim, N., Nasser and T., sheltami, “A Fault Tolerant Dynamic Clustering Protocol of wireless sensor networks”, IEEE communications, 2009.

[29] A., Khadivi and M., shiva, “FTPASC: a Fault Tolerant Power Aware protocol with static Clustering for wireless sensor networks”, IEEE, 2006.

[30] Y., Lai and H., chen, “Energy-efficient Fault Tolerant Mechanism for Clustered Wireless Sensor Networks”, IEEE, 2007.

[31] D.Y., Kim and J., Cho, “Active Caching: A transmission Method to Guarantee Desired Communication Reliability”, In wireless sensor networks. IEEE, 2009.

[32] M. N. Cheraghlou, S. Babaie and M. Samadi, “LRC: Novel fault tolerant local re-clustering protocol for wireless sensor network”, J. Computer, 4 (2012) 2151–9617.

[33] S., Gobriel, S., Khattab, D., Moss, J., Brustoloni, and R., Melhem, “RideSharing: Fault Tolerant Aggregation in Sensor Network using Corrective Actions”, In Proceeding of Third Annual IEEE Communications Society Conference on Sensor, Virginia, (September 2006).

[34] S., Ratnasamy, B., Karp, S., Shenker, D., Estrin, R., Govindan, L., Yin, and F., Yu, “Data-Centric Storage in Sensornets with GHT”, a Geographic Hash Table. Mob. Netw. Appl., 8(4):427-442, August 2003.

[35] I., Saleh, M., eltoweissy, A., ahbariya and H., el-sayed, “A Fault Tolerance Management Framework for wireless sensor networks”, Journal of communications, vol.2, no.4, june2007.

[36] M. N., Cheraghlou, A., Khademzadeh and M., Haghparast, “Increasing lifetime and fault tolerance capability in wireless sensor networks by providing a Novel Management Framework”, Wireless Press Communication, 04 Aug 2016.

[37] H., taheri and H., Boroumand, “Proposing a Method to Increase Fault Tolerance in Wireless Sensor Networks Using Fuzzy Logic”, International Journal of Computer & Information Technologies (IJOCIT), 2015, pp. 691-699.

[38] S., Behzadi, M., Azad, “Fault-tolerant in Wireless Sensor Networking using fuzzy logic”, International Research Journal of Applied and Basic Sciences, Vol.8, no.9, 2014, pp. 1276-1282.

[39] S. J., Dastgheib, H., Oulia, H., Gholamishiri and A., Ebrahimi, “A Proposed Fuzzy Method to Detect Faulty Nodes in Wireless Sensor Network Clustering Structure”, International Journal on Technical and physical Problems of Engineering, Vol.4, no.2, 2012, pp. 71-75.

[40] A., Farzadnia, A., Harounabadi and M. M. Lotfinejad, “A Fuzzy Method for Fault Tolerance in Mobile Sensor Network”, Journal of academic and Applied Studies, Vol. 2(12), 2012, pp. 52-62.

[41] S., Chang and T., Huang, “A Fuzzy Knowledge Based Fault Tolerance Algorithm in Wireless Sensor Networks”, 2012 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 26-29 March 2012, pp. 891-896.

[42] P., Chanak,I., Banerjee, T., Samanta and H., Rahaman, “FFMS: Fuzzy Based Fault Management Scheme in Wireless Sensor Networks”, Volume 305 of the series Communications in Computer and Information Science, 2012, pp. 30-38.

[43] B., Moses Sathyaraj, L. C., Jain, A., Finn and S., Drake, “Multiple UAVs path planning algorithms: a comparative study”, Fuzzy Optimization and Decision Making, September 2008, 7:257.

[44] H., Lan, and Juan J. Nieto, “Solving implicit mathematical programs with fuzzy variational inequality constraints based on the method of centres with entropic regularization”, Fuzzy Optimization and Decision Making, December 2015, Volume 14, Issue 4, pp. 493-511.

[45] H., Lee and S., Guu, “On the Optimal Three-tier Multimedia Streaming Services”, Fuzzy Optimization and Decision Making, March 2003, Volume 2, Issue 1, pp. 31-39.

[46] M.N., Cheraghloua, A., Khadem-Zadehb and M., Haghparast, “A survey of fault tolerance architecture in cloud computing”, Journal of Network and Computer Applications, Vol.61, February 2016, Pages 81–92.

Mehdi Nazari Cheraghlou received his B.Sc. degree in computer hardware engineering from Azad University, in Tehran. In 2010 he received his M.Sc. degree in computer architecture from Tabriz Branch, Azad University. He is currently Ph.D. student in Tehran Branch of Azad University. His research fields are optics and optoelectronics designs, reversible logic gate and quantum computing. His Current research interests focus on design target management models, fault tolerance architectures and frameworks for wireless sensor networks and sensor cloud computing.

Ahmad Khadem-Zadeh received the B.Sc. degree in applied physics from Ferdowsi University, Mashhad, Iran, in 1969 and the M.Sc. and Ph.D. degrees respectively in Digital Communication and Information Theory & Error Control Coding from the University of Kent, Canterbury, U.K. and He was the head of Education & International Scientific Cooperation Development, the head of Test Engineering Group and the director of Computer and

Communication Department at ITRC. He is also a lecturer at Tehran Universities and is a committee member of Iranian Computer society and also a committee member of the Iranian Electrical Engineering Conference Permanent Committee. Prof. KhademZadeh has been received four distinguished national and international awards including Kharazmi International Award, and has been selected as the National outstanding researcher of the Iran Ministry of Information and Communication Technology.

Majid Haghparast received his B.Sc. in computer hardware engineering in 2003 and his M.Sc. and Ph.D. degrees in computer architecture in 2006 and 2009, respectively. Since 2007, he has been affiliated with the Computer Engineering Faculty, Yadegar-e-Imam Khomeini (RAH) Shahre Rey Branch, IAU University, Tehran, Iran. His research interests include cloud computing and computer arithmetic. Since April 2017 he is conducting his sabbatical at the Johannes Kepler University Linz, Austria, where he also is a Research Fellow. Dr. Haghparast is on the panel of reviewers for various international journals.