International Journal of Computer Applications (0975 – 8887)
Volume 143 – No.11, June 2016

Use of Contextual Information to Optimize Wireless Sensor Node's Energy Consumption Rate during Reprogramming Procedures

Eronu Majiyebo Emmanuel
Electrical/Electronics Engineering Department
Faculty of Engineering
University of Abuja, Abuja, Nigeria

Ibrahim Tashiwa Emmanuel
Electrical/Electronics Engineering Department
Faculty of Engineering
University of Abuja, Abuja, Nigeria

ABSTRACT

This research work explores the use of context information to improve upon wireless sensor networks performance by reducing the energy consumption rate during reprogramming processes. A software system that dynamically reconfigures wireless sensor network operational functionalities optimally based on evolving application context is developed. In order to demonstrate the benefits of the context based reconfiguration model, two contexts related input variables were used. The first variable is obtained using a metric tool (PDE) devised for extracting the delta of two files (application related context). The second variable entails the battery energy level state of the sensor node taken as an operational-demand related context. A robust inference engine was developed based on the inferred expert knowledge on memory related energy consumption pattern during the reconfiguration process. The resulting output from the fuzzy logic controller decides when and which of the reconfiguration approach is to be implemented in order to prolong the battery life. The model's performance was evaluated on an OMNet++ simulation platform using pilot data obtained from a test bed composed of Microchips' PIC32MX320F128H microcontroller and MRF24J40MB transceiver. In a network of six nodes, two were equipped with the developed model capability and the others were not. The overall energy expended as read, erase and write were obtained from each node for the purpose of comparison. Results obtained show that 65% of energy expended during the erasure procedure is saved in nodes that adopt the context-based reconfiguration model. Similarly, 45% and 69% reduction in energy consumption were obtained for the read and write procedures respectively.

Keywords

Reprogramming, wireless sensor network, reconfiguration, fuzzy logic controller, Algorithm.

1. INTRODUCTION

Sensor network application can be expensive to implement, especially when large-scale projects are involved. Being able to manage network resources and tailor their use towards several other applications other than what they were initially designed for can be a daunting task. Application objectives, anticipated constraints, resource managerial strategies and other surrounding factors, when well spelt out in the design model, simplify the complexity arising from adapting WSN to newer applications. Identifying these factors requires a careful analysis of the entire wireless sensor network (WSN) operational environment. When these factors are considered as a source of relevant contextual information, then reconfiguring WSN becomes much easier.

Baldauf, Dustdar, and Rosenberg in [1] described context-aware systems as systems that can alter their mode of operation to suit the current context without explicit user intervention thereby increasing the systems usability and effectiveness. Context awareness is commonly used in systems whose operation or responses are influenced by certain defined surrounding factors. The concept of context-aware systems allows applications to gather context data and adapt their operational behaviour accordingly. These applications can function without explicit intervention and thereby increase their usability and effectiveness within the context of the environment where they operate [1]. Context-driven allows a system to assign resources on current and relevant tasks, rather than just processing predefined applications. Equipping the node with relevant context sensing capabilities enables it to estimate future context requirements. When these requirements are used appropriately, the network can be configured to perform more optimally. Management systems can guess about what kind of tasks will be required in the near future and consider it when allocating resources.

Hence, effecting the sensor nodes' reconfiguration processes based on contextual information can be helpful in several ways. For example, deciding on when and how to effect a reconfiguration process can result in reducing the system's operational cost. This cost invariably entails energy consumed and memory size utilised by the nodes during the reconfiguration process.

2. REVIEW OF RELATED LITERATURE

Steine *et al.*, in [2] introduced an approach that exploits design-time knowledge of the application scenario dynamics to construct and implements a proactive runtime reconfiguration paradigm. However, two challenging issues are apparent here: the possibility of capturing all anticipated reconfiguration needs can be challenging, and the scarcely available memory space might not be sufficient to accommodate codes written to address these needs. Moreover, even if it does, there is the likelihood of redundant codes written to handle anticipated changes, which might never occur, and invariably taking up scarcely available memory spaces.

In addition, implementing WSN reconfiguration may depend on whether it is needful, urgent, or sustainable. For example, instead of effecting reconfiguration procedure during unfavourable weather conditions that have negative effect on transmission signals, it may be needful to delay the process and then resume when the conditions become favourable. In extreme cases, it is advisable to stop the process completely when the available energy in the node cannot sufficiently sustain the reconfiguration process. Where the second option is the norm, the sensor node might not be able to implement new functionalities but it can still be utilised for other purposes not dependent on the update. The ability to take decisions of this nature is largely confined to the human domain. However, Artificial Intelligence (AI) techniques like the Fuzzy Logic and Artificial Neural Network allow machines to mimic human cognitive capabilities. Importantly, the problem needs to be presented as defined input variables and the output variables make-up the solutions. Solutions are obtained from the analyses of processed input variables in conformity with a set of rules that are based or derived from expert knowledge.

In view of these needs and observed deficiencies in existing approaches [2, 4] this research work is intended to address the following:

- Reduce the presence of redundant codes, thereby lessening the size of the firmware deployed to the wireless sensor nodes;
- Enhance the flexibility of reuse; allow real-time user input during reconfiguration processes and autonomous reconfiguration using fuzzy logic in decision making;
- Establish a two-way interactive platform between the reconfiguring agent (user via base station) and the reconfigured (sensor node) and by extension the entire wireless sensor network. The two-way interactive platform enables the base station to assess the state of the sensor node through the contextual information it relayed. In addition, coupled with other relevant information (operational related contextual information), the system then decides when and how or what manner of reconfiguration should be employed. The aim is to ensure that the entire network performs efficiently and optimally manages the available resources (memory usage and energy consumption)
- To integrate and use fuzzy logic controller in deciding the most appropriate reconfiguration approach to adopt in response to evolving application or operational context.

3. DESIGN CONSIDERATIONS

In this section, some of the background information that guides our decisions in determining dominant independent variables which affect energy consumption pattern within the WSN reprogramming context are presented and discussed.

3.1 Flash Memory Reprogramming Constraints

Three possible reconfiguration scenarios are highlighted in Figure 1, Figure 2 and Figure 3. In each Figure, two columns of a set of blocks designated as ‘$SegO_0$…$SegO_n$’ and ‘$SegN_1$…$SegN_n$‘ represent original and reconfigured contiguous segments of flash memory respectively. Reconfigured data are represented by a strip of filled rectangular blocks. As shown in Figure 1, the first scenario describes a situation where the number of reconfigured data bytes is confined to a single segment ‘$SegN_1$’. In such a scenario, erasure and rewriting procedures should naturally be limited to a single segment. However, in practice, this is not always the case; the entire flash memory is always erased, and the new firmware rewritten all again. The repeated occurrence of the erasure and rewriting procedures will eventually accelerate energy consumption at a higher rate. The second scenario as depicted in Figure 2 illustrates the space taken in memory by the reconfigured data.

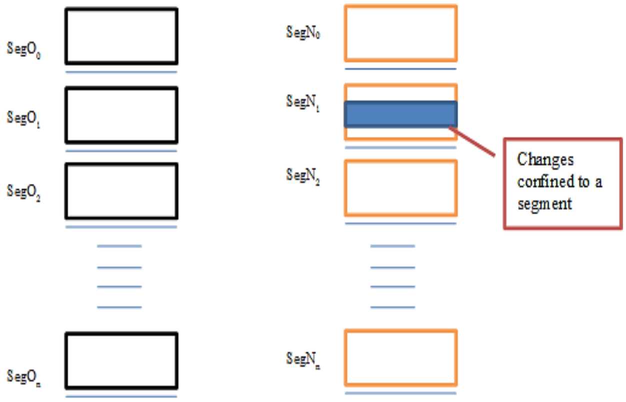


Fig 1: Reconfigured Data confined to a single segment

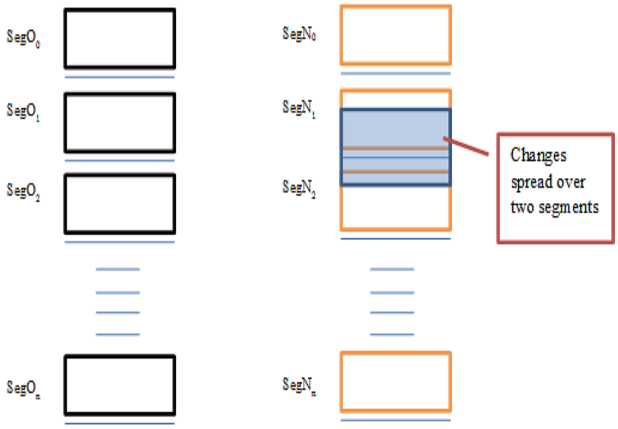


Fig 2: Reconfigured Data Spread Over Adjoining Segments

The space overlaps adjoining segments and being able to handle erasure and writing operations within these two segments will invariably result in consumption of much less energy. The third scenario, shown in Figure 3, depicts changes in the new firmware that are unevenly distributed all over the memory space. This is attributed to changes resulting from the addition, removal or renaming of functions within application source codes. These can be more complex when the functions are referenced in several places inside the application source code. Similar problems exist for global data variables [5, 6].

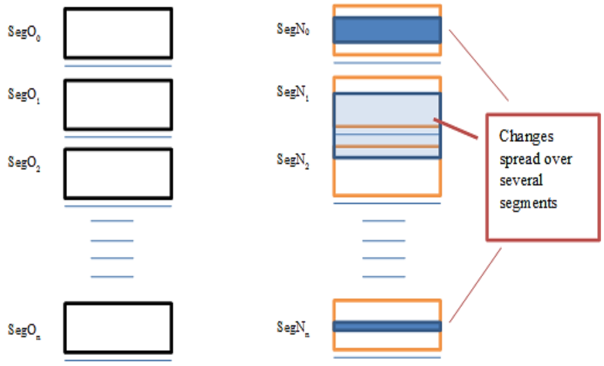


Fig 3: Changes spread over several segments

3.2 Firmware Reconstruction Algorithm

In cases where changes occur within a single segment and considering the ‘erase before rewriting’ constraint associated with Flash memories, it is economical to ensure that the reconstruction procedures are confined to that segment. Based on the data obtained from earlier works [7, 8], it can be inferred that the cost of erasing an entire memory is far less than erasing individual segments. Likewise, writing to a segment is much cheaper than writing to each word that makes of a segment. The norm in practice has been to erase the entire Flash memory and then reprogram it with the new update. Based on the analysis highlighted in the preceding section, three delta-orientations were inferred. These are namely ‘Segment-confined’, ‘Adjoint-Segments’ and ‘Disjoint-Segment’. In practice, only the first and the last are more pronounced. A delta extraction tool based on the Precise Delta Extraction (PDE) scheme as presented in [*3*] provides the address of every delta detected, which invariably can be helpful in pinpointing the exact segment where they occur. This information allows for erasure and rewriting operations to be carried out within only selected segment(s) of relevance.

Algorithm 1 listing highlights the re-flashing algorithm developed and employed in the context-based WSN reconfiguration software system. $\boldsymbol{a_k}$ and $\boldsymbol{d_k}$ represent the address and data of delta extracted by the PDE utility where k signify the index or position of each member in the set with cardinal value of m. Let $\boldsymbol{SO_i}$ and $\boldsymbol{SN_j}$ denote segments containing the original and modified firmware in flash memory where i and j are their respective locations within a set of n segments contained in the flash memory. $\boldsymbol{T(r)}$ connotes an array for storing the index or indices of segment(s) affected by the modifications or reconfigurations.

Algorithm 1: Flash Program Memory re-flashing

```
$r = 0; j = 1; k = 1$
While ( $j<=n$) do
    While ($k<=m$) do
        If ($a_k$ => start address of $SN_j$ & $a_k$ <= start address of $SN_j$)
            $T( r ) = j$
        end if
        $k++ ; r++ ; j++$
    end while
end while
Select |$T( r )$|
Case 1:
    Erase and reprogram within $SO_{T(0)}$
Case 2:
    Erase and reprogram $SO_{T(0)}$ and $SO_{T(1)}$
Case >2:
    Erase and reprogram entire memory space
end select
```

4. METHODOLOGY

The benefits of the context based reconfiguration model is demonstrated via the use of two contexts related input variables. The first variable is obtained using a delta extraction tool refer to as Precise Delta Extraction (PDE)[3] devised for extracting context information from the delta of two files (application related context). The second variable entails the battery energy level state of the sensor node taken as an operational-demand related context. A robust inference engine was developed based on the inferred expert knowledge on memory related energy consumption pattern during the reconfiguration process. The pattern studied and presented explains how delta size and its orientation can influence energy consumption while reprogramming sensor nodes

4.1 Test Bed Hardware and Software Composition

A test bed made-up of an ad hoc network of three 32-bit processor based sensor nodes. It is meant to provide some pilot data for evaluating the context based reconfiguration software at a much larger scale using the OMNeT++ and Castalia WSN simulation platform (http://omnetpp.org/). The Test Bed features a powerful Microchip PIC32MX320F128H microcontroller (http://ww1.microchip.com/) and a Microchip MRF24J40MB transceiver (http://ww1.microchip.com/) for implementing low-cost Wireless Sensor Network. Figures 4, 5 and 6 depicts an overview of the Testbed hardware composition, the base station and a single Sensor Node and an attached transceiver respectively. The embedded software system implemented in each source code runs on the Contiki operating system platform (https://contiki-os.org).

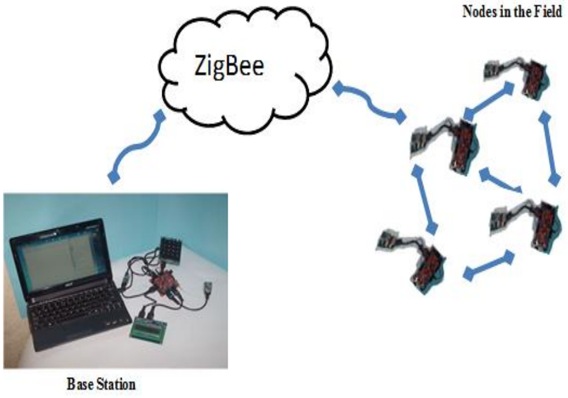


Fig 4: The Test Bed Hardware Composition

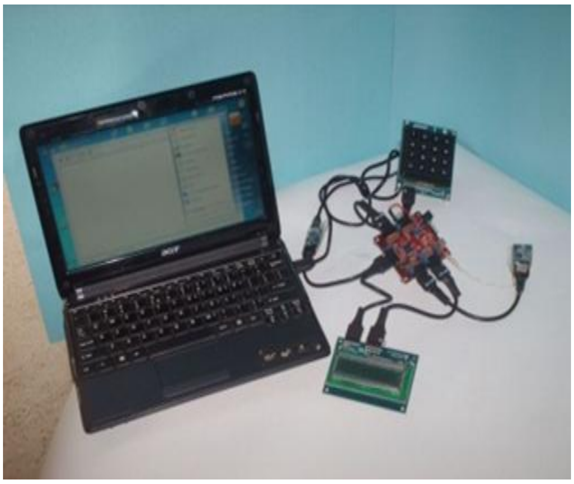


Fig 5: Base station Hardware Composition

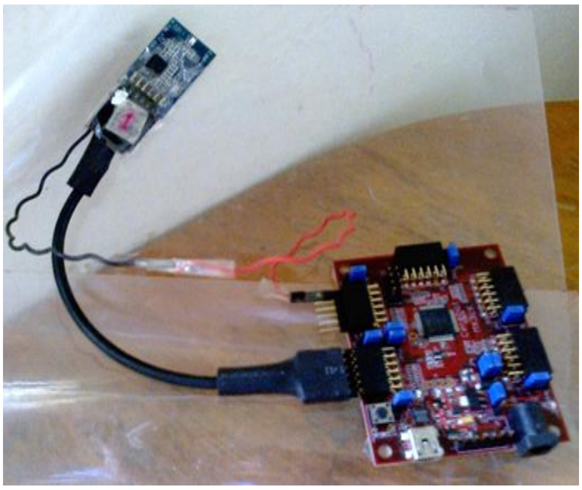


Fig 6: WSN Hardware with an attached transceiver

4.2 Design and Implementation of a Fuzzy Logic Control Component

In developing a fuzzy logic control component, use is made of the Fuzzylite fuzzy logic controller [9]. In addition, it features an application named qtfuzzylite which allows for visual design and real time interactions with the FLCs.

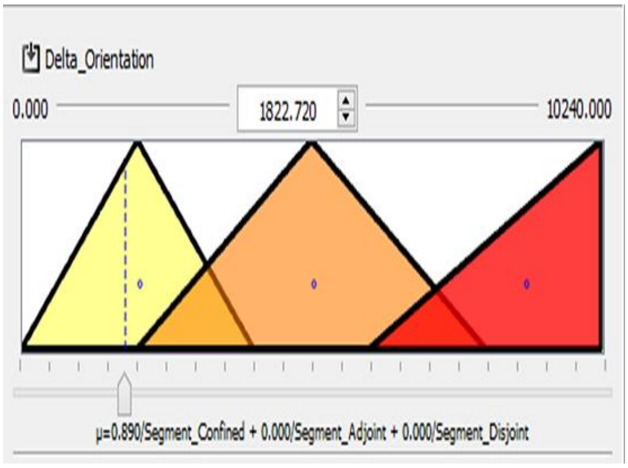
4.2.1 Modelling the System's Inputs and Output

The qtfuzzylite is employed to define the structure of the controller by creating the most suitable linguistic variables and membership functions for the application. The triangular membership functions were adopted because they have been used extensively in real-time applications due to their simple formulas and computational efficiency [10].

The Delta-orientation obtained via the PDE and the sensor node's battery energy state served as input into the fuzzy logic system. The delta-orientation and the battery energy state were meant to represent the application and operational-demand context respectively.

The input membership functions shown in Figure 7 are defined for the Delta orientation input. It takes into account the three possible delta-orientations presented in section 3, these are as follows: Segment-confined, Segment-Adjoint and Segment-Disjoint. The delta-orientation entails three membership functions spread over a range of 2.5 * number of bytes contained in the segment of program memory (for the PIC32MX320F128H, each segment contains 4096 bytes).

The second input value for the fuzzy-logic system is the battery energy state expressed in terms of joules. As shown in Figure 8, the range of this input value is spread over the values of 0 to 18720 Joules, where 18720 Joules is the typical energy of two AA batteries (http://castalia.npc.nicta.com.au). The range maps the sensor node's battery energy level between when it is in a virtually depleted state to a fully charged state.

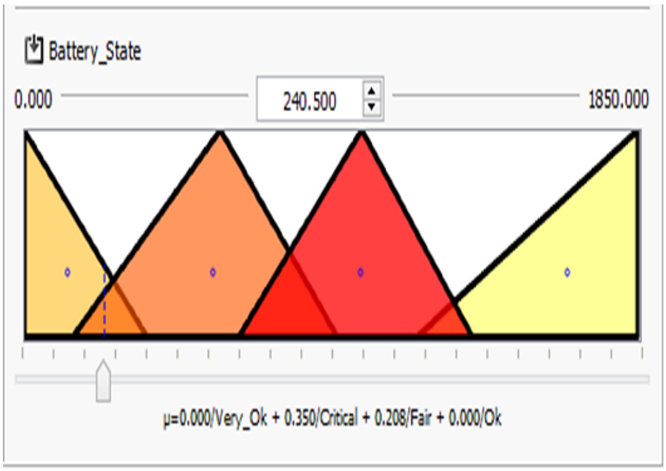


Delta_Orientation

0.000 1822.720 10240.000

μ=0.890/Segment_Confined + 0.000/Segment_Adjoint + 0.000/Segment_Disjoint

Fig 7: Membership function for Delta Orientation input value



Battery_State

0.000 240.500 1850.000

μ=0.000/Very_Ok + 0.350/Critical + 0.208/Fair + 0.000/Ok

Fig 8: Membership functions for Battery energy state input value

The membership functions of the battery energy state input were distributed as follows:

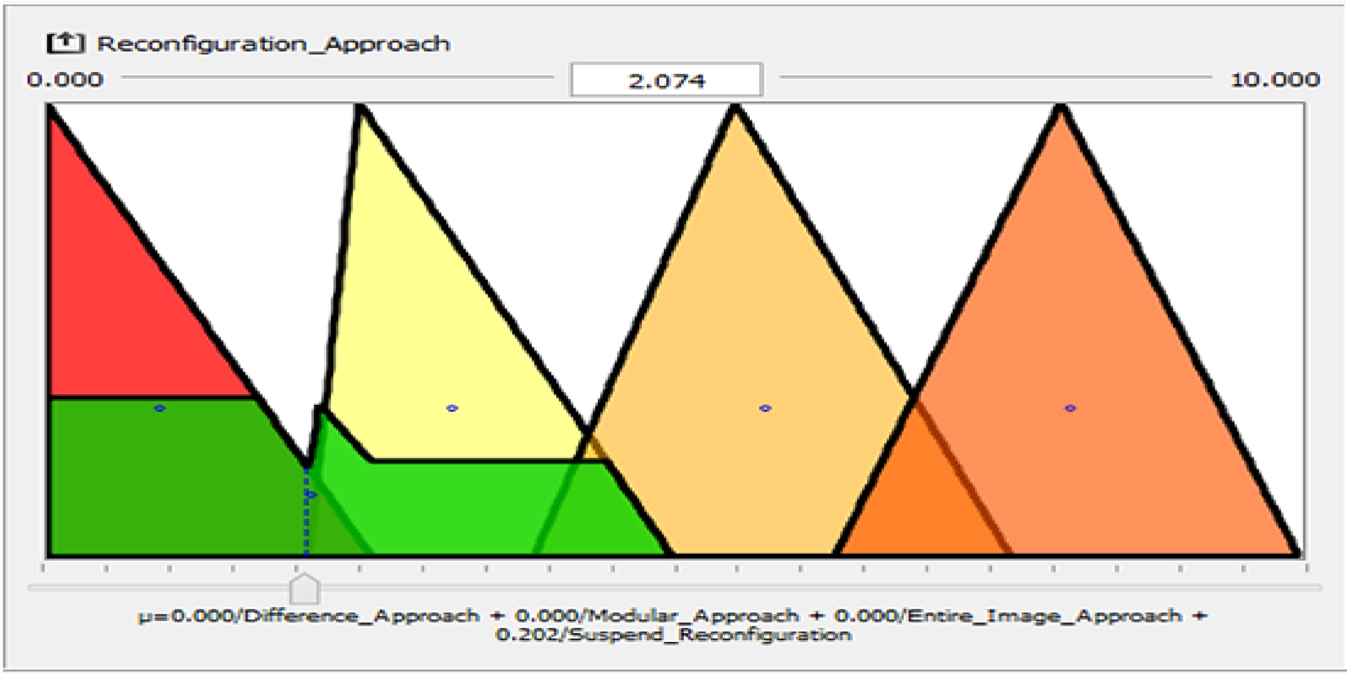
i. Critical: Cannot support reconfiguration for any delta size or orientation; energy should rather be conserved for application's basic task

ii. Fair: Can support reconfiguration if delta size is within an acceptable size range- most probable a segment.

iii. OK: Can support whole or any delta size of reconfiguration. However, should be used with caution.

iv. Very OK: More than sufficient energy is available to handle any delta size or orientation.

The fuzzy system calculates the fuzzy-cost for each Delta and battery energy state input values. The ensuing output membership function intended to guide each sensor node in adopting the most appropriate reconfiguration approach while considering the available battery's energy level is shown in Figure 9. The distribution is spread over four options: Difference-Approach, Modular-Approach, Entire-Image-Approach, and Suspend-Reconfiguration. This process applies to each wireless sensor node while in the field. This ensures that the battery's energy level in every node is optimally managed during every reconfiguration process.



μ=0.000/Difference_Approach + 0.000/Modular_Approach + 0.000/Entire_Image_Approach + 0.202/Suspend_Reconfiguration

Fig 9: Output membership function intended to guide each sensor node in adopting the most appropriate reconfiguration approach

4.2.2 *Fuzzy Inference Engine*

The fuzzy inference engine is composed of rules developed using expert knowledge. The design of the knowledge-based rules that connect the inputs, and the outputs is based on the philosophy behind reprogramming of Flash memory. This philosophy has been discussed in section 3.

The fuzzy inference system is designed based on twelve rules listed below:

i. if Delta_Orientation is Segment_Confined and Battery_State is Very_Ok then Reconfiguration_Approach is Difference_Approach;

ii. if Delta_Orientation is Segment_Confined and Battery_State is Critical then Reconfiguration_Approach is Suspend_Reconfiguration;

iii. if Delta_Orientation is Segment_Confined and Battery_State is Fair then Reconfiguration_Approach is Difference_Approach;

iv. if Delta_Orientation is Segment_Confined and Battery_State is Ok then Reconfiguration_Approach is Difference_Approach;

v. if Delta_Orientation is Segment_Adjoint and Battery_State is Very_Ok then Reconfiguration_Approach is Modular_Approach;

vi. if Delta_Orientation is Segment_Adjoint and Battery_State is Critical then Reconfiguration_Approach is Suspend_Reconfiguration;

vii if Delta_Orientation is Segment_Adjoint and Battery_State is Fair then Reconfiguration_Approach is Modular_Approach;

viii if Delta_Orientation is Segment_Adjoint and Battery_State is Ok then Reconfiguration_Approach is Modular_Approach

ix. if Delta_Orientation is Segment_Disjoint and Battery_State is Very_Ok then Reconfiguration_Approach is Entire_Image_Approach;

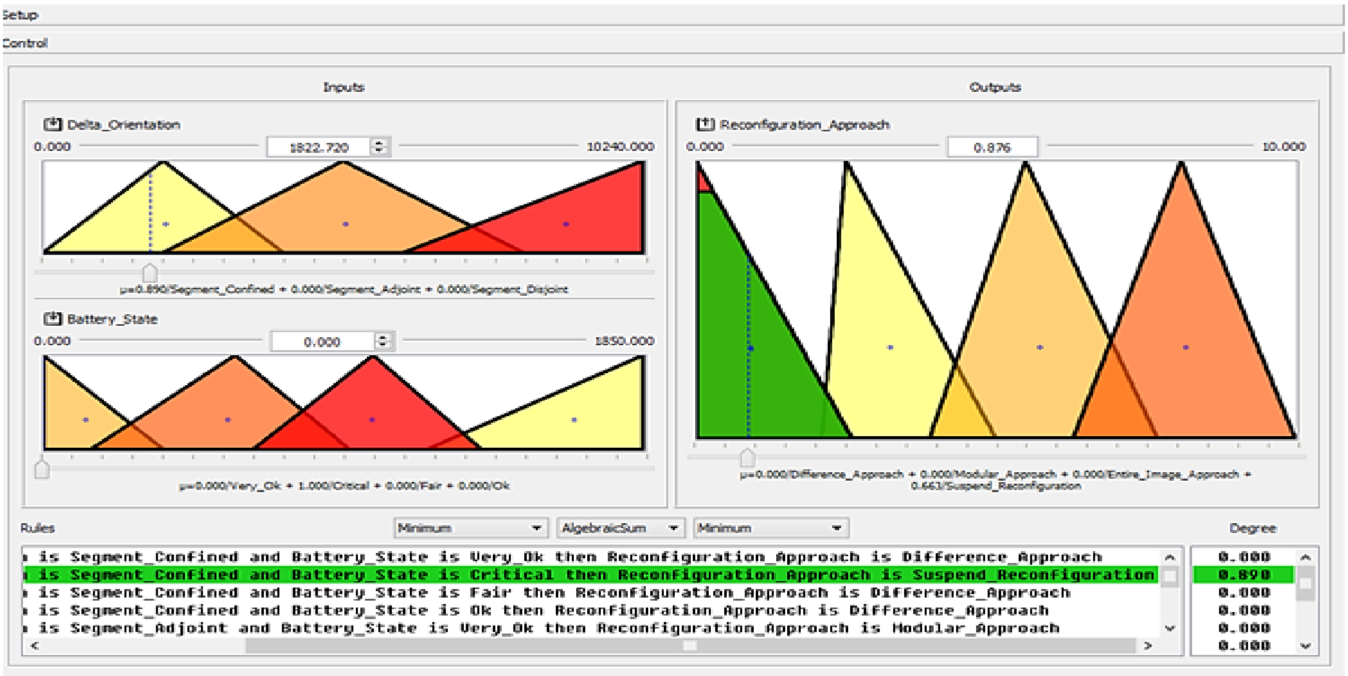
x. if Delta_Orientation is Segment_Disjoint and Battery_State is Critical then Reconfiguration_Approach is Suspend_Reconfiguration;

xi. if Delta_Orientation is Segment_Disjoint and Battery_State is Fair then Reconfiguration_Approach is Suspend_Reconfiguration; and

xii. if Delta_Orientation is Segment_Disjoint and

Battery_State is Ok then
Reconfiguration_Approach is
Entire_Image_Approach.

The qtfuzzylite development tool’s rule text editor offers an easy way to examine and define the set of rules. Using these features, one can verify that all the defined rules are necessary, that no important rules are missing, and that the variations of the output variable are consistent with the designed system requirements. Optimising the entire system (Figure 10) behaviour is done easily and quickly by changing the set of rules, modifying the membership functions definitions, or selecting from the available defuzzification options.



Delta_Orientation: 0.000 — 1822.720 — 10240.000
μ=0.890/Segment_Confined + 0.000/Segment_Adjoint + 0.000/Segment_Disjoint

Battery_State: 0.000 — 0.000 — 1850.000
μ=0.000/Very_Ok + 1.000/Critical + 0.000/Fair + 0.000/Ok

Reconfiguration_Approach: 0.000 — 0.876 — 10.000
μ=0.000/Difference_Approach + 0.000/Modular_Approach + 0.000/Entire_Image_Approach + 0.661/Suspend_Reconfiguration

Rules: Minimum, AlgebraicSum, Minimum

```
 is Segment_Confined and Battery_State is Very_Ok then Reconfiguration_Approach is Difference_Approach      0.000
 is Segment_Confined and Battery_State is Critical then Reconfiguration_Approach is Suspend_Reconfiguration 0.890
 is Segment_Confined and Battery_State is Fair then Reconfiguration_Approach is Difference_Approach         0.000
 is Segment_Confined and Battery_State is Ok then Reconfiguration_Approach is Difference_Approach           0.000
 is Segment_Adjoint and Battery_State is Very_Ok then Reconfiguration_Approach is Modular_Approach          0.000
                                                                                                            0.000
```

Fig 10: Complete Fuzzy control Platform

4.3 Simulation Platform

The simulation platform adopted for the purpose of evaluating the performance of the complete system is the Castalia based on the OMNeT++. Castalia is a simulator for WSN and networks of low-power embedded devices. It is used to test distributed algorithms and/or protocols in realistic wireless channel, with a realistic node behaviour especially relating to access of the radio.

Adopting the Castalia framework, a network of six wireless sensor nodes was setup on the OMNeT++ platform. One of the nodes *SNode[0]* serves as the agent that links all the other sensor nodes to the base station, three of the nodes *SNode[2]* , *SNode[3] and SNode[5]* were programmed with context based reconfiguration capabilities and the remaining other two nodes *SNode[1]* and *SNode[4]* take on default reconfiguration paradigm. Figures 11 depicts the nodes arrangement within the network. The OMNeT++ simulation kernel records the message exchanges during the simulation into an *event log file*. This log file can be analysed later with the Sequence Chart tool. The Sequence Chart tool (Figure 12) shows how the message is routed between the different nodes in the network. The sequence chart is valuable for debugging, exploring or documenting the complex model behaviour.

The dataset obtained from the PDE and the Fuzzy controller sub components were used to run the simulation platform. Erasure, reading and writing energy, as well as the total energy consumption resulting from both erasure and writing operation were compared for the two sets of nodes. These results and the ensuing discussion are presented in section 5.

4.4 Adopted Data for Evaluation Purpose

A summary of the results obtained for three case studies described in [*3*] and adopted for this research work are presented in Table 1. The results obtained were categorised under the following: the delta(s) size, the physical address range of the delta(s), related ELF segments where the delta resides, delta orientation and the number of segment(s) involved.

Table 1. A summary of the results obtained for three case studies adopted for this research [3]

Case study	Description of Application Change Scope	Size of changes in byte	Physical Address Range(s)		ELF segment Name	Orientation of Change in Memory(Delta Orientation)	Number of Segment
			Start	End			
1	Effecting Changes to 'Constant' Data	2	9D012014	9D012014	.text	Segment confined	1
2	Effecting Changes to 'Flow of Control	3	9D00F280	9D00F2C8	.text	Segment confined	1
3	Effecting Changes to 'Function's Name'	2725	9D000028 9FC01280 A00025FC BFC00014 BFC02FF0	9D013884 9FC01984 A0002784 BFC00194 BFC02FF0	.text .vector .data .reset .config_BFC02FF0	Segments Disjointed	5

5 EVALUATION PROCEDURES

5.1 Fuzzy Inference Engine

The fuzzy inference engine's performance is critical to achieving the set goals of every context based reconfiguration system for WSN. The details of its design have been discussed in section 4.2. In order to demonstrate the designed fuzzy inference system's application and performance, two simulated scenarios were used.

The first scenario is based on case study one (1) as presented in Table 1 Here, the delta size of two (2) bytes is defined to belong to the segment-confined membership function (delta-orientation). Varying the battery's energy level over the designated ranges as shown in Figure 11 results in the 'Difference-approach' reconfiguration option being suggested (though of a degree of 0.001, even when the battery's energy level is at a critical level).

The second scenario is based on case study three. The delta size obtained as indicated in Table 1 is 2725, though the size is a single segment range; the delta's orientation is of the disjointed nature. Figures 12, Figures 13 and Figures 14 shows the various reconfiguration options selected based on varied battery's energy levels. Table 2 shows the results obtained when the battery's energy level was varied across its selected corresponding membership functions.

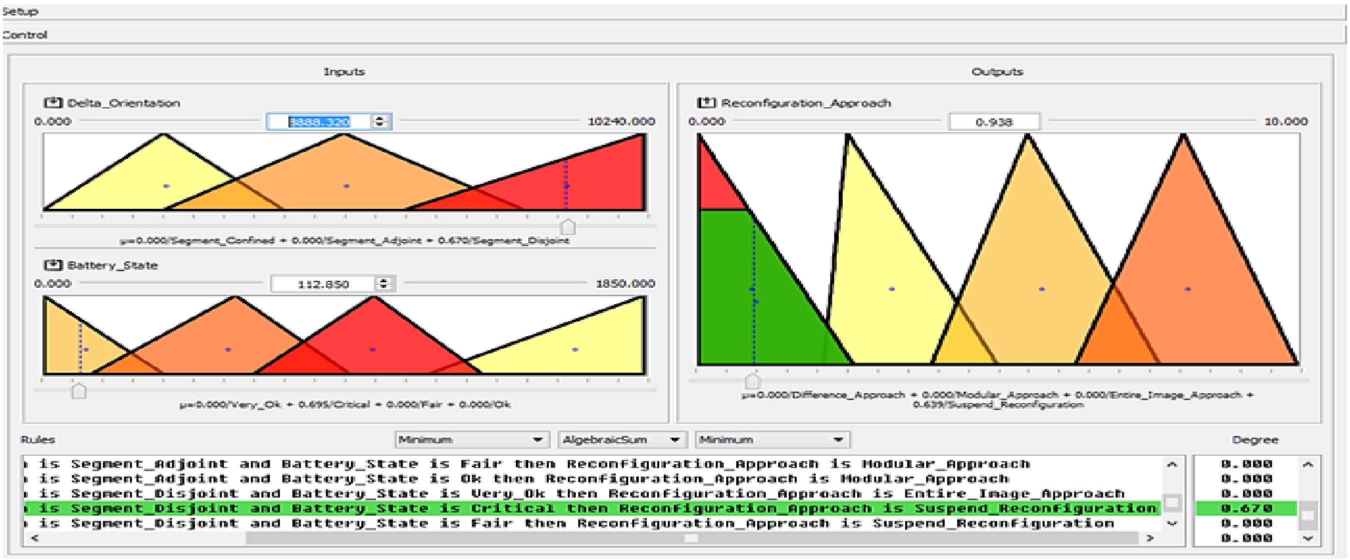
Table 2. Results obtained for varied battery's energy levels

SN	Battery state / Degree	Reconfiguration Option / Degree	Best Reconfiguration option/ Degree
1	Critical / 0.695	Suspend reconfiguration / 0.670	Suspend reconfiguration / 0.670
2	Ok / 0.639 Fair / 0.168	Suspend reconfiguration / 0.168 Entire-Image-Approach / 0.639	Entire-Image-Approach / 0.639
3	Very Ok / 0.758	Entire-Image-Approach / 0.670	Entire-Image-Approach / 0.670

```
μ=0.000/Segment_Confined + 0.000/Segment_Adjoint + 0.000/Segment_Disjoint
μ=0.000/Very_Ok + 0.000/Critical + 0.792/Fair + 0.000/Ok
μ=0.580/Difference_Approach + 0.000/Modular_Approach + 0.000/Entire_Image_Approach +
0.000/Suspend_Reconfiguration
```

```
 is Segment_Confined and Battery_State is Very_Ok then Reconfiguration_Approach is Difference_Approach     0.000
 is Segment_Confined and Battery_State is Critical then Reconfiguration_Approach is Suspend_Reconfiguration 0.000
 is Segment_Confined and Battery_State is Fair then Reconfiguration_Approach is Difference_Approach         0.001
 is Segment_Confined and Battery_State is Ok then Reconfiguration_Approach is Difference_Approach           0.000
 is Segment_Adjoint and Battery_State is Very_Ok then Reconfiguration_Approach is Modular_Approach          0.000
                                                                                                             0.000
```

Fig 1: Test demonstration based on case study 1



```
μ=0.000/Segment_Confined + 0.000/Segment_Adjoint + 0.670/Segment_Disjoint
μ=0.000/Very_Ok + 0.695/Critical + 0.000/Fair + 0.000/Ok
μ=0.000/Difference_Approach + 0.000/Modular_Approach + 0.000/Entire_Image_Approach +
0.639/Suspend_Reconfiguration
```

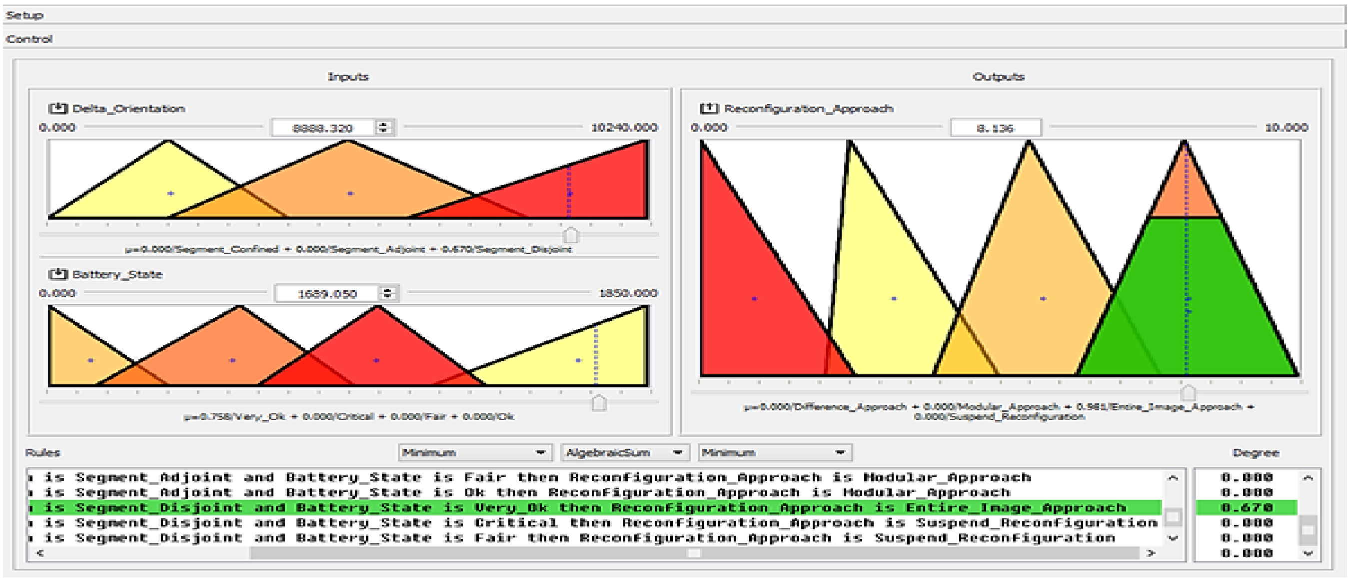
```
 is Segment_Adjoint and Battery_State is Fair then Reconfiguration_Approach is Modular_Approach               0.000
 is Segment_Adjoint and Battery_State is Ok then Reconfiguration_Approach is Modular_Approach                 0.000
 is Segment_Disjoint and Battery_State is Very_Ok then Reconfiguration_Approach is Entire_Image_Approach      0.000
 is Segment_Disjoint and Battery_State is Critical then Reconfiguration_Approach is Suspend_Reconfiguration   0.670
 is Segment_Disjoint and Battery_State is Fair then Reconfiguration_Approach is Suspend_Reconfiguration       0.000
                                                                                                               0.000
```

Fig 2: Test demonstration based on case study 3 for battery state set as critical

```
μ=0.000/Segment_Confined + 0.000/Segment_Adjoint + 0.670/Segment_Disjoint
μ=0.000/Very_Ok + 0.000/Critical + 0.168/Fair + 0.639/Ok
μ=0.000/Difference_Approach + 0.431/Modular_Approach + 0.251/Entire_Image_Approach +
0.000/Suspend_Reconfiguration
```

```
 is Segment_Adjoint and Battery_State is Ok then Reconfiguration_Approach is Modular_Approach                 0.000
 is Segment_Disjoint and Battery_State is Very_Ok then Reconfiguration_Approach is Entire_Image_Approach      0.000
 is Segment_Disjoint and Battery_State is Critical then Reconfiguration_Approach is Suspend_Reconfiguration   0.000
 is Segment_Disjoint and Battery_State is Fair then Reconfiguration_Approach is Suspend_Reconfiguration       0.000
 is Segment_Disjoint and Battery_State is Ok then Reconfiguration_Approach is Entire_Image_Approach           0.168
                                                                                                               0.639
```

Fig 3: Test demonstration based on case study 3 for battery state changing from Ok to fair



```
is Segment_Adjoint and Battery_State is Fair then Reconfiguration_Approach is Modular_Approach      0.000
is Segment_Adjoint and Battery_State is Ok then Reconfiguration_Approach is Modular_Approach        0.000
is Segment_Disjoint and Battery_State is Very_Ok then Reconfiguration_Approach is Entire_Image_Approach   0.670
is Segment_Disjoint and Battery_State is Critical then Reconfiguration_Approach is Suspend_Reconfiguration   0.000
is Segment_Disjoint and Battery_State is Fair then Reconfiguration_Approach is Suspend_Reconfiguration   0.000
                                                                                                    0.000
```

Fig 4: Test demonstration based on case study 3 for battery state set to Very Ok

5.2 The Complete System

Each of the sample files' source codes was altered or modified in response to changes emanating from evolving application needs. Typically, these changes could involve or span over variables, constants, function names, libraries and other source code constructs. However, in this work the changes were confined to variation involving constants, variables and Function names only. Having implemented these changes, the modified files were then recompiled to obtain new ELF files. Each pair of generated ELF files (original and modified) were further processed using the PDE. The PDE, by design, outputs a dataset, which contains a collection of delta (the data difference(s) between the original and modified files) and their respective address or addresses where applicable.

Adopting the Castalia framework, a network of six wireless sensor nodes is setup on the Omnet++ platform (illustrated in Figure 15). One of the nodes *SNode [0]* is positioned to serve as the reconfiguration agent. The agent routes both Data and control messages from the base station to the other nodes (*SNode [1]*, *SNode [2]*, *SNode [3]*, *SNode [4]* and *SNode [5]*) in the network. Three of the nodes *SNode [2]*, *SNode [3]* and *SNode [5]* were programed with the context based reconfiguration capabilities and the remaining other two nodes *SNode [1]* and *SNode [4]* take on default reconfiguration paradigm. The set is equipped with context based reconfiguration capabilities is tagged as group A while those with default reconfiguration paradigm tagged as group B.

In order to evaluate the benefits of the context-based reconfiguration model, each node in the simulation setup is configured to take on default values of energy consumed per byte and per segment during program memory reprogramming operations (read, erase and write procedures). The intent is to ascertain whether there is a significant difference in the amount of energy consumed by the two set of nodes. The parameters used in configuring each node are listed in Table 3. These parameters were used in computing the read, erasure and write energy consumption values in the Omnet++ simulation platform. The values were adopted from the literature [7, 11, 12] and the datasheet of the test bed's microcontroller (PIC32MX320F128H).

The simulation procedure involves the transmission of a packet of data consisting of delta and control messages from the base station to each of the nodes (*SNode [1]*, *SNode [2]*, *SNode [3]*, *SNode [4]* and *SNode [5]*) via *SNode [0]* as shown in Figure 15.

The sequence chart shown in Figure 16 illustrates the pattern of transmission and reception as implemented within the WSN. In addition, the sequence chart presents the history of the simulation carried out. The control message was derived from the output of the Fuzzy logic controller. The integration of the fuzzy logic inference engine into the Omnet++ simulation platform was implemented via the use of fuzzilite dataset generated using the qtfuzzilite tool.

Using a test delta size of 2725, delta orientation of the segment-confined type and battery energy state of 'very ok', the read, erasure, write and the total energy (a summation of read, erasure and write energy values) consumed were obtained and subsequently used to plot the graph shown in Figure 17 in section 5. Similarly, using a test delta size of 2725, delta orientation of the segment-disjoint type and battery energy state of 'very ok', the read, erasure, write and the total energy (a summation of read, erasure and write energy values) consumed were obtained and subsequently used to plot the graph shown in Figure 18 in section 5.

Table 3. Flash Memory Characteristic [7]

Procedure Scope	**Energy(μ J)**		
	Read	**Write**	**Erase**
Per Byte	0.004	0.009	0.047
Segment/Page	0.0679	7.66	192.2

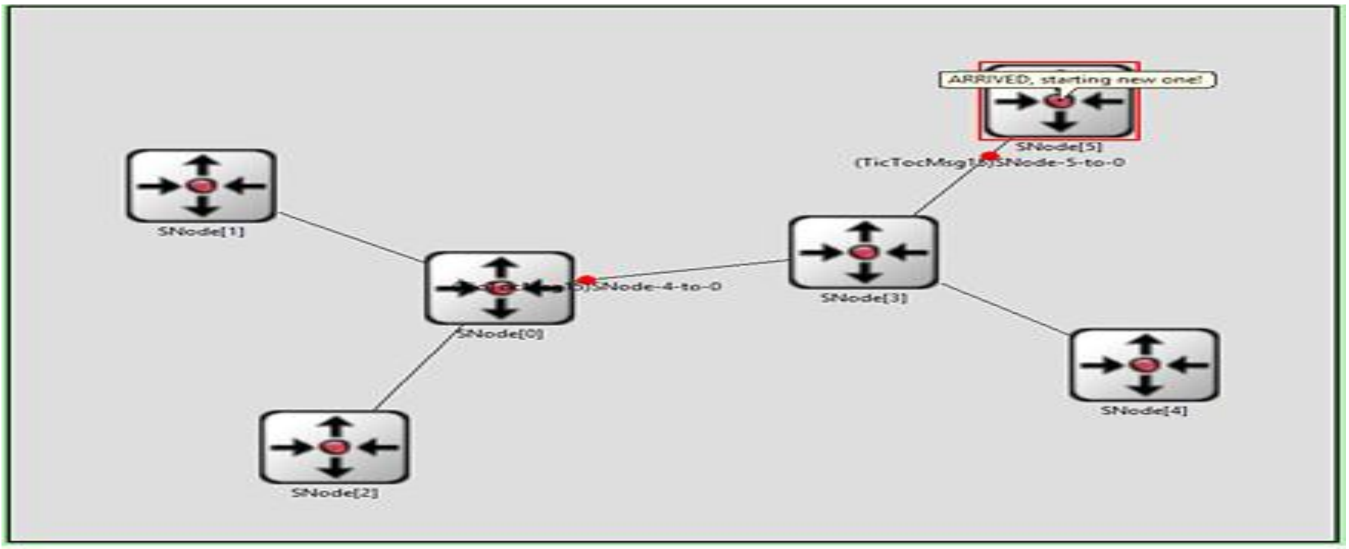


Fig 5: Network Composition in Omnet++

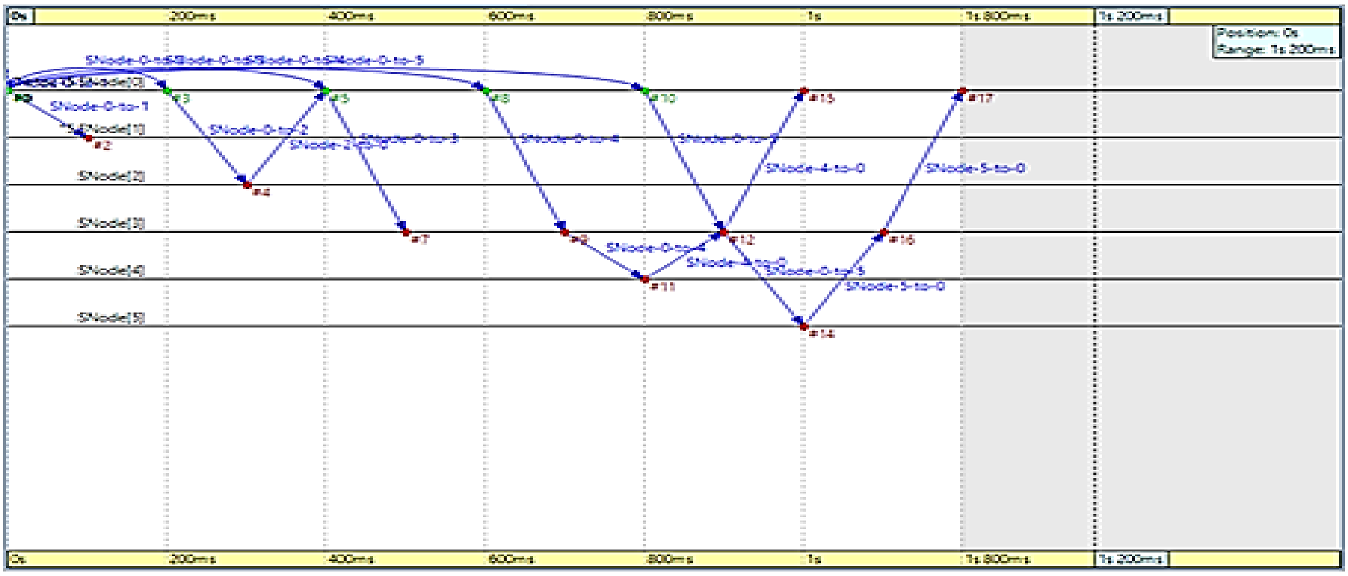


Fig 6: The Sequence Chart tool shows how the message is routed between the different nodes and the history of the simulation carried out

6 RESULTS AND DISCUSSIONS

The graph shown in Figure 17 was obtained for group A set of nodes where the delta orientation is of the segment-confined type. In conventional reprogramming procedures, the entire program memory is erased and rewritten all over again with a new image even if the delta (change) is a minute fraction of the entire program memory space. Hence the result obtained in Figure 18, also represents what is obtainable for the second group of nodes (group B) for the delta-orientation set as segment-confined. However, when the delta orientation is of the segment-disjoint type and irrespective of what the delta size is, both groups A and B set of nodes adopt the conventional reconfiguration approach. Therefore, the results obtained are similar to that indicated in Figure 18.

Comparing the two graphs indicates that 65% of energy expended during the erasure procedure is saved when the context based reconfiguration model is adopted. Similarly, 45% and 69% reduction in energy consumption were obtained for the read and write procedures respectively. The implication is that quite a considerable amount of energy is wasted when very minute deltas with segment-confined orientation are involved.

Additional contextual information are applicable. For example, the signal strength of each sensor node may vary over space and time. These can negatively affect the reconfiguration process especially where retransmission occurs several times due to poor signal reception occasioned by poor weather conditions. In such situation, the norm is to stop reconfiguration completely. However, in a context based reconfiguration approach, if the delta detected is relatively small that it can be handled with much less resources expended, then the reconfiguration process is allowed to take place.

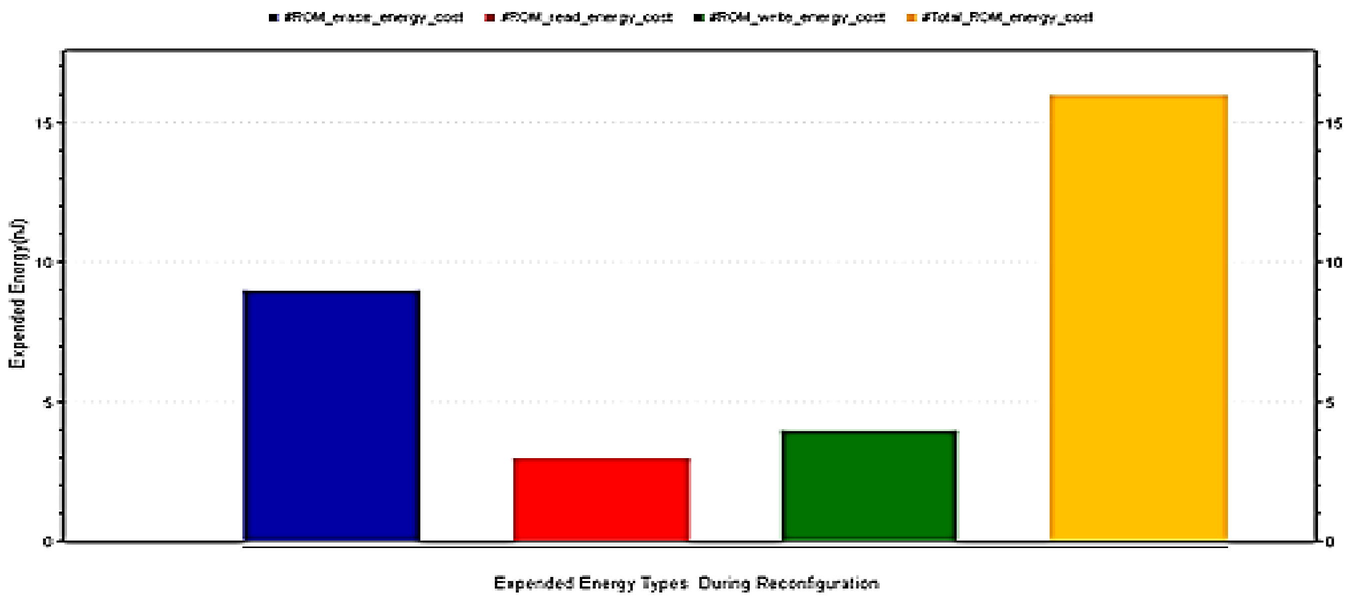


Fig 7: Graphical plot made for deltas' size less than program memory's segment Size, having segment-confined orientation type

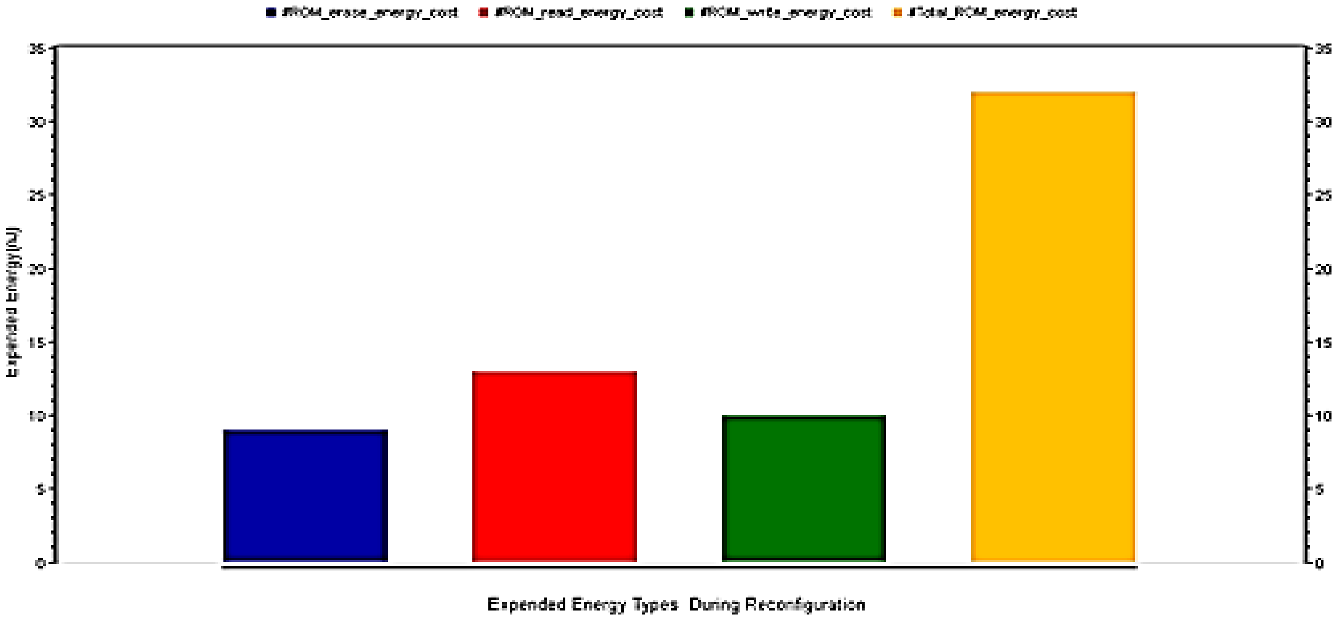


Fig 8: Graphical plot made for deltas' size less than the program memory's segment size, having segment-disjoint orientation type

7 CONCLUSIONS

This work demonstrates the implementation and benefits of adopting a context based reconfiguration model in optimising energy consumption during WSN reprogramming. Two contexts related input variables were used. The first variable is obtained using a metric tool (PDE) devised for extracting the delta of two files (application related context). The second variable entails the battery energy level state of the sensor node taken as an operational-demand related context. A robust inference engine was developed based on the inferred expert knowledge on memory related energy consumption pattern during the reconfiguration process. In a network of six nodes, two were equipped with the developed model capability and the others were not. The overall energy expended as read, erase and write were obtained from each node for the purpose of comparison. The results obtained show that about 65% of energy expended during the erasure procedure is saved in nodes that adopt the context based reconfiguration model. Similarly, 45% and 69% reduction in energy consumption were obtained for the read and write procedures respectively.

8 REFERENCES

[1] Baldauf, M., Dustdar, S., and Rosenberg, F. 2007. "A Survey on Context-aware systems". International Journal of Ad Hoc and Ubiquitous Computing, 2(4), 263-277.

[2] Steine, M., Ngo, C.V., Oliver, R. S. Geilen, M., Basten, T., Fohler, G. and Decotgnie J. 2011. "Proactive Reconfiguration of Wireless Sensor Networks", Proceedings of the 14th ACM International Conference on Modelling, Anaylsis and Simulation of Wireless and Mobile systems (MSWIM'11), 31-39.

[3] Eronu, E.M., & Misra, S. (2016). "Precise Delta Extraction Scheme for Reprogramming of Wireless Sensor Nodes". Nigerian Journal of Technology (NIJOTECH). 35(1), 144-154.

[4] E. M. Eronu, S. Misra and M. Aibinu, 2013 "Reconfiguration Approaches in Wirless Sensor Network: Issues and Chanllenges," in 2nd IEE Conference on Emerging & Sustanable Technologies for Power & ICT in a Developing Society (NIGERCON), Owerri, Nigeria.

[5] Dong, W., Chen, C., Bu, J. and Huang, C. 2013. "Enabling efficient reprogramming through reduction of executable modules in networked embedded systems", Journal of Ad Hoc Networks. 11(1). 473-489, doi>10.1016/j.adhoc.2012.07.007

[6] Dong, W., Liu, Y., Chen, C., Bu, J., Huang, C. and Zhao, Z. 2011. "R2: Incremental Reprogramming using Relocatable Code in Networked Embedded Systems". IEEE INFOCOM conference proceedings, 376 – 380.

[7] Han-Lin, L., Chia-Lin, Y. and Hung-Wei, T. 2008. "Energy-Aware Flash Memory Management in Virtual Memory System", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 16(8), 952-964.

[8] Gaurav, M., Peter, D., Deepak, G. and Prashant, S. 2006. "Ultra-low power data storage for sensor networks", Proceedings of the 5th international conference on Information processing in sensor networks, April 19-21, Nashville, Tennessee, USA. Doi:10.1145/1127777.1127833.

[9] Rada-Vilela, J. 2014. "Fuzzylite: a fuzzy logic control library", URL: http://www.fuzzylite.com.

[10] Sadiq, A. S., Abu, B. K. and Ghafoor, K. Z. 2010. "A Fuzzy Logic Approach for Reducing Handover Latency in Wireless Networks" . Network Protocols and Algorithms, (2)4, 1 -27, doi: 10.5296/npa.v2i4.527

[11] Mohan, V., Bunker, T., Grupp, L., Gurumurthi, S. and Stan, M. R. 2013, "Modeling Power Consumption of NAND Flash Memories Using FlashPower", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 32(7), 1031-1043.

[12] Mathur, G., Desnoyers, P., Ganesan, D. and Shenoy, P. 2006. "Ultra-low power data storage for sensor networks", Proceedings of the 5th international conference on Information processing in sensor networks, April 19-21, Nashville, Tennessee, USA , doi:10.1145/1127777.1127833.