A Fault Tolerance Fuzzy Knowledge Based Control Algorithm in Wireless Sensor Networks

Shih-Hao Chang, Ping-Tsai Chung, Teh-Sheng Huang

Abstract

In this paper, we present a new knowledge based control algorithm for fault detection and diagnosis and recovery using fuzzy logic approach and challenge approach namely Fault Tolerance Fuzzy Knowledge based control in Wireless Sensor Networks (WSNs). This FTFK algorithm diagnoses faulty communication between sensor nodes providing Fault Detection and Isolation (FDI) to eliminate faulty communication behaviour nodes in the WSNs communication framework. Our algorithm also monitors erroneous behaviour of nodes to ensure that faulty nodes sharing incorrect data under certain qualities can be tolerated in the network. We have conducted a simulation study for this FTFK algorithm using the Georgia Tech Network Simulator (GTNetS). Our simulation results show that FTFK scheme can improve performance and reliability of WSNs: it expenses little bit higher energy and examine time to achieve shorter average end to end delay and better packet success ratio when a number of cluster heads failures in the WSNs.

Keywords: Wireless Sensor Networks, Network Management, Fuzzy Knowledge Based Control, Fault Tolerance, Network Reliability, Fault Diagnosis and Isolation, Byzantine Behavior.

1. Introduction

Recently Micro-Electro-Mechanical System (MEMS) technology rapid development, which allows us to build Wireless Sensor Networks (WSNs) led to the possibility of combining sensors, RF transceivers and processing capability into small, low-power nodes. WSNs is a network made up of a number of such nodes that can now be easily deployed in a wide variety of environments, making them very attractive for large-scale applications like environmental monitoring, security surveillance, and disaster relief. On the other hand, WSNs introduces new challenges for fault-tolerance. Due to sensor nodes normally deployed a large number of information in the hostile or ill-fitting environment; consequently, it is very common that these low cost of sensor nodes may result to arbitrarily faulty behaviour. Moreover, whether sinks will be able to receive the sensing data are strongly dependent on the reliability of network.

Fault-tolerance is a significant concern in designing telecommunication systems and is becoming vitally important for designing WSNs as it has ability to self-configure, self-adapt desired level of functionalities in the presence of faults. To achieve fault tolerance design in WSNs, the key is to develop a maturity fault tolerance by the Fault Detection and Isolation (FDI) techniques. The environment of WSNs is dynamic, a specific functionality of FDI can not deal with a large variety of process variables generated by multiple issues (such as hardware, software, ...), therefore, it makes very difficult for the network managers to diagnose fault process and then take appropriate actions to effectively perform the tasks of FDI. Fuzzy Knowledge Based Control [14, 15] is an interesting research area based on fuzzy logic and its objective is to model human reasoning by using a Knowledge-Based System (KBS) including a knowledge base and an inference engine. FTFK is based on a fuzzy logic approach that can be realized by a KBS for performing a dynamic and intelligent system control. FTFK has been applied in wide range of applications, including flight control, robot mobility behaviour, ... etc. It is suitable for modelling FDI by applying FTFK algorithm implementing by a KBS to help in making decision.

Byzantine Fault Problems (BFPs) has been addressed in [1] as “one in which a component of some system not only behaves erroneously, but also fails to behave consistently when interacting with multiple other components.” Lamport offered an elegant solution to this problem by detection of up to
m traitors given $3m + 1$ or more generals [1]. Since the sensor is typically placed in unfitted locations and can suffer malicious attackers, to insure the data correctness in WSNs is an extremely important issue. To avoid incorrectness data fusing and prevent arbitrary nodes behaviour affecting data correctness, we follow the consensus based algorithm [16] to solve fault tolerance in BFPs. In this paper, we introduce a FTFK algorithm, and a KBS to provide an intelligent fault tolerance model and to detect Byzantine faults on clustered hardware in WSNs. The rest of the paper is organized as follows. In Section II, background and related works are briefly described. In Section III, we classify the sources of the faults. In Section IV, we present the FTFK algorithm and propose a data reliability mechanism to solve fault detection and diagnosis in WSNs. Our simulation study will be present in Section V. Finally, our conclusions and future work are explored in Section VI.

2. Background & Related Works

Currently Byzantine tolerance solutions cause redundancy in computation and retrofitting. Lamport [1] was the first one who described the Byzantine Generals Problem (BGP) and apply to the distribute computer systems. BGP was stated in [1] that there are n generals of Byzantine army with their divisions, some of them are traitors. These generals have to an agreement on deciding a war mission of attack or retreat. Because of these generals are geographically separated and they only can communicate with one another by messengers. Lamport’s concern in [1] was “how to create a consensus among a number of independent processors which allows the fault-free processors to agree on a common value in a distributed system”. Lamport [1] tried to offer an elegant solution to this problem by detection of up to f traitors given $3f + 1$ or more generals and must exists recursive algorithm for all generals to exchange message with each other. However in [1], they proved a complicated construction solution, this fault preventing algorithm requires network synchronization and it used a complicated construction which is hard to implement in WSNs.

Byzantine fault for Mobile Sinks Wireless Sensor Networks (MSWSNs) were described in [2], there are two approaches have been proposed to combat the Byzantine faulty nodes for data gathering in MSWSNs. In [4], Mogilevsky et al. tried to solve the mobile agent routing problem employing an evolutionary multi-objective optimization algorithm. In order to compute a mobile agent routing in faulty nodes environment, it is usually very involved because it requires to trade-offs among the computation energy consumption, routing path, packet losses, and the accuracy of detection. Mogilevsky et al. [4] utilized randomized censored averaging (RCA) scheme and randomized median filtering (RMF) scheme to eliminate the effect of faulty sensors for data gathering and to provide a fault tolerance routing in the mobile agent based sensor network. Note that the faulty nodes may be due to erroneously behaviours from hardware failures or software failures in WSNs, in which sensor nodes normally deployed in the hostile environment.

Aucsmith [3] proposed a self-checking technology in which embedded code segments verify the integrity of a software program as the program is running. They partitioned code segments are encrypted and are handled in a fashion such that only a single segment is ever decrypted at a time. These embedded code segments check that a running program has not been altered, even by one bit. However, to perform the checking mechanisms security using this approach depends heavily on the procedure and the software detection program. On the other side, neighbour monitoring in all its aspects has become more crucial in recent years. A neighbour monitoring could be used in terms of detecting neighbour node’s arbitrating behaviour affecting data collection and message communication.

Koushanfar and Potkonjak [6] developed a family of Markov Chain-based models for identification of faulty data reading for widely used MICA2 sensor motes. They adapted a class of markov chain models called Semi-Markov Chain models that ensure the correct lagged autocorrelation statistical properties, while keeping size of the models very compact. This model provides better protocols for collecting data in the presence of faulty and missing samples. However, there are many limitations for the semi-markov chain models of the instrumented environment such as these models do not complete statement of faulty situations, recovery mechanism and do not concern the redundant nodes or coverage issue to insure network connectivity.

Kim [7] proposed a Supervisor-based Network Surveillance (SNS) scheme to effectively monitor a variety of neighbour node behaviour in point-to-point networks environment. This SNS scheme using neighbour node as a work node to observe the healthy condition of it neighbour nodes and to report a fault suspicion reports to the supervisor node. The supervisor node using collected information to judge
whether a fault has indeed occurred, and then sends the fault occurrence notice to all the healthy
worker nodes in the system. This SNS scheme is close to our consensus based checking algorithm.

As we discussed in Section 1, the FTFK algorithm is to model human thinking processes and is
implemented by a KBS for various applications. The cluster head in our fault model will broadcast
challenges messages to its vicinity cluster heads. Each healthy cluster head receive this challenge
message will execute the challenge command and response answer message to present its correctness.
The FTFK algorithm is discussed in detail in the Section 4.

This FTFK algorithm combine the consensus based algorithm and fuzzy knowledge to provide
adaptive thresholds to evaluate and diagnoses faulty nodes in WSNs. This algorithm mainly focuses on
how to eliminate faulty nodes behaviour to maintain network operation in the WSNs. The main
contribution of the second tier fault tolerance algorithm is to replace the $\lambda_j$ by the $\kappa$ parameters.
The healthy parameter $\kappa$ is to express quantitatively the goodness of source data which apply the consensus
based fault detection mechanism using adaptive thresholds. Our key contribution is providing a general
diagnostic framework that distinguishes sensor nodes problems so that the correct troubleshooting
operation(s) can be performed.

3. Failure Sources

Each sensor node in WSNs consists of components of sensing, data processing, communications and
power modules. Factors, such as software failures, or hardware malfunctions will attract and harsh
WSNs environment which may further result in system failures in WSNs. Therefore, system stability
of sensor node is one of critical issues in our fault diagnosis mechanism. Moreover, sensor nodes
shared wireless communication medium to each other which may cause noise interference, message
drop and link temporary failure, therefore, communication reliability of sensor node is another of
critical issue to reach the goal. Besides, cluster head is required to fuse the collected sensing data from
neighbouring sensor nodes before they are sent to the sink. For this reason, information correctness of
the data fusion is the other critical issue in our fault tolerance. As the consequences, we categorized
into three failure sources: System Failure, Communication Failure, and Arbitrariness Fault.

System Failure: The correctness of the sensing data is strongly depends on the reliability of sensor
system. The Mean Time Between Failure (MTBF) is a measure typically be represented the reliability
of a system to perform its required functions under stated conditions for a specified period of time. The
lower the MTBF number means the lower reliability of the system. We concern system failure rate
during the normal operation life and we measure the probability of the failure. The faulty node found
by a self-checking algorithm or a software code which checks its system metrics. Once the probability
of the failure of this node has over its system metrics threshold, it will force a system rebooted or
shutdown and no further communications are required. Otherwise, a recovery algorithm can be
performed and it takes a brief period of time to recover the fault. Sometimes, if a system was not
successfully recovered from a system failure, it may suffer even more severe impacts. In this case, we
assume that the system needs to be rebooted with longer average duration. Unsuccessful recovery is
most commonly caused by error propagations when the effect of a failure is not sufficiently shielded
from the rest of the system. Failed sensors need to be repaired, with the mean time to repair (MTTR). If
the failure sensors is not recovered correctly, failure sensors should be accommodated by replacing the
redundant or neighbouring sensors. Moreover, these failures sensors may be caused by either system
failure or energy expenditure and these failures sensors will result in a negative impact on network
operations, including the sensing coverage of the area and the connectivity among nodes.

Communication Failure: The network connectivity between the target source nodes and sink(s) is
the most critical issue, and it should be kept as good as possible during the entire network lifetime. The
communication failure is introduced by sharing the wireless channels causing noise interferences,
packet losses which are resulted in a negative impact on the network connectivity. Currently researches
on maintaining network connectivity for good reputation networks are generally relying on the
Bayesian networks [17]. Bayesian networks are combining statistical data with prior knowledge about
the problem domain, which makes a reputation network. As networks become more complex and
dynamic, it’s a challenging work to maintain a sufficient level of expertise on a particular network’s
behaviour. Remedial network connectivity will probably involve replacing or fixing a node or the
network path in its vicinity.
**Arbitrariness Fault:** Once the system and communication fault sources have been classified, we concern the data reliability problem in WSNs. To avoid faulty sensors impact to the reliability of result sensing data, minimizing the impact of faulty sensor measurements is related to the Byzantine Fault Problem (BFP). Byzantine fault nodes may deviate arbitrarily without any reason and can send arbitrary values to other neighbouring component during collaboration. The data sent from neighbouring component is possibly inaccurate and may affect the correct data available with the valid component. In order to accommodate BFP, Mogilevsky et al. [4] implemented cluster based consensus-checking scheme in parallel programming systems to provide FDI scheme. In [5], Gupta, and Younis, implemented challenge-response and consensus-checking by the initiator cluster head to solve Byzantine faulty problem (BFP). The vicinity healthy nodes would respond by sending the checking results back to the initiator that achieves fault detection.

4. **Solution Model**

4.1. **Intelligent fault tolerance agent**

In this section, we describe an intelligent fault tolerance agent that prevents malfunction of the system, unreliable communication link and malicious data which impact the performance of WSNs. This intelligent fault tolerance agent learns the normal behaviour of each measurement variable and applies the FTFK algorithm to provide fault detection and isolation. The diagram of independent fault detection agent module is illustrated in Fig. 1, it contains the power supply, sensing unit, memory, microcontroller, transceiver, power management module and fault tolerance agent.

![Independent Fault Detection Agent Module](image)

**Figure 1. Independent Fault Detection Agent Module**

4.2. **Fuzzy knowledge based control**

Fuzzy Knowledge Based Control (FTFK) is based on fuzzy logic approach that consists of a four-step process: Fuzzification, Knowledge-Based Processing, Defuzzification and Post Processing. Firstly, the residuals data have to be fuzzified, then they have to be evaluated by a knowledge base, and finally they have to be defuzzified. The transformation of an objective term into a fuzzy concept is called fuzzification. The fuzzification is based on membership functions to mapping of the residuals in the form of linguistic variable derived. These membership functions are formulas used to determine the fuzzy set to which a value belongs and the degree of membership in that set. The rule based matched with the specific linguistic data from above rule involves (using fuzzification) looking up the Membership Value (MV) of the inputs. To translate back fuzzy concepts into objective terms which can be used in practice, defuzzification is to place the membership value generated by rule based for each fuzzy outcome and calculated to generate the appropriate output. Finally, in the post processing stage, we will validate range of estimated outputs extracted fuzzy rules and provide knowledge based to control the current fault situation.

4.3. **Fault detection and diagnosis**

The fundamental principles of diagnostic of sensor system and communication fault are based on the diagnosis of residual data processing. Generally speaking, FTFK algorithm is recollecting of human thinking processes and use linguistic variable to nonlinear mapping of input data vector into a symptom. In our FTFK algorithm, fault diagnosis is strongly depending on adaptive thresholds to evaluate
residuals data in WSNs. Because FTFK algorithm is based on fuzzy logic which should be satisfied the following three assumptions:

(i) The sensor node must be continuously monitored to acquire the communication characteristic signals.
(ii) Residuals must be available, this residuals analytical functions which are fed by monitoring outputs.
(iii) Suitable threshold values have to be gathering to evaluate for all the residuals.

First, let us consider the following deterministic model which describes the process under normal operating conditions, where B is the output measurement vector, u is the input vector and x represents the state vector.

\[ A = V_m (u, x); \quad B = V_s (u, x) \]  

Second, the residuals are analytical functions based on the continuous comparison of the redundancy data from system that is collected in the same conditions. This redundancy data comes from the processing. The fault decision making process is to evaluate the residuals with predefined thresholds. Three common approaches to generate predefined threshold values are listed as follows:

(i) Quantities value checking: When system model is composed of measurable quantities values.
(ii) Output signals comparison: Output signals are compared to the nominal estimated model of the system.
(iii) Output parameters comparison: Output parameters are compared to the actual model of the system.

Suppose a residual \( r(u, y) \) has been generated with the aid of a diagnostic observer and \( y \) is the output measurement vector. For fault detection the residual must meet the following conditions:

\[ r(u, y) = 0 \text{ if no fault presents} \]  
\[ r(u, y) \neq 0 \text{ if a fault presents} \]

Third, suitable threshold values have to be determined through experimental or simulation tests. It has to be noted that threshold values mainly depend on measurement uncertainty and modelling errors, and strongly influence the fault detection. The optimal choice of the detection thresholds depends on these uncertainties and varies with the operating, therefore, a different approach has been presented in [8], [9]. Sauter et al. [10] introduced the idea of adaptive threshold which consists of choosing an optimal threshold according to the operating conditions. This adaptive threshold will decide the fault signal or data which is dependent on these uncertainties and varies with the operating conditions.

**Fuzzification**: To understand the FTFK algorithm, we explain the basic principle in terms of the fuzzification in FTFK. Once the residuals were received, the measurements of residuals are normally superimposed by noise. Since all these disturbances affect the residual, it is necessary to assign thresholds larger than zero in order to avoid false alarms. In order to avoid this shortcoming, suppose the predefined threshold can be “softened” by splitting it up and tolerate a small disturbance false alarm. We can insert an interval of a finite step to change the false alarm tendency. The effect of a finite step cause results in false alarms is avoided by such a softening of the threshold. By composition of the fuzzy sets \{zero\} and \{one\} a membership diagram for the residual is obtained as shown in Fig.2. Evidently, the fuzzification of a threshold can directly be interpreted as a fuzzification of the residual.

**Figure 2.** Composition of the fuzzy sets \{zero\} and \{one\}

**Figure 3.** Representation of Fault Decisions Using a Fault Tree
Rule-based: General specking, the task of fault decision is to infer \( \mathbf{f} \in \mathbf{F} \) of the set \( \mathbf{F} \) of possible faults from a set \( \mathbf{R} \) of residuals, where \( \mathbf{r} \in \mathbf{R} \). In our case, the residuals \( \mathbf{r} \) are defined by their fuzzy sets and the relationships between the residuals and the faults are given IF-THEN rules. For example:

\[
\begin{align*}
\text{IF} & \quad (\text{faulty element B}), \\
\text{THEN} & \quad (\mathbf{r}_1 \text{ middle or large} \quad \text{AND} \quad (\mathbf{r}_2 \text{ are small}) \quad \text{AND} \quad (\mathbf{r}_3 \text{ are small})
\end{align*}
\]

The residuals \( \mathbf{r} \) evaluating these rules, it becomes possible under certain conditions to find for each combination of residuals corresponding with a fault [7, 14]. In other words: The fuzzy conditioned statement is a mapping of the residuals onto the faults with the aid of the rules in the knowledge base. This can be illustrated by directed graphs or fault-trees as shown in Fig. 3. Since the rules are of the form the way through the fault tree is directed against the direction of the causality, the connections between the faults to the residuals (effects, such as large, middle and small) constitute a crisp mapping.

Defuzzification: In this stage in the fuzzy control procedure that allows the transition from the symbolic domain to the numerical one. This means that the fuzzy information about the faults has to be converted to yes-no decisions. Defuzzification provides a crisp value that best represents a single or composite fuzzy set. Several defuzzification algorithms exist, like centre of area (gravitycenter), firstmaximum, meanofmaximum, etc. To decide the threshold problem in FTFK, we apply the adaptive thresholds [15]. Let \( J_0 \) be the nominal threshold computed under nominal operating conditions \( (u_0, x_0) \). The resulting relation for the fuzzy threshold adaption is given by

\[
J_0 = J_0 (u_0, x_0)
\]

And \( J_0 \) is the threshold used for detection whatever update the detection operating conditions are. We suggest a threshold according to:

\[
J(u,x) = J_0 (U_0, X_0) + \Delta J (U-U_0, X-X_0)
\]

where \( \Delta J \) represents an increase or a decrease according to the operating conditions and is computed with the aid of fuzzy rules.

Post Processing: Symptoms are then generated from comparing the residuals to the updated thresholds: larger is the difference between each residuals and its threshold, higher is the possibility that a fault has occurred.

Existing literature in WSNs have proposed solutions to resolve each of the individual problems in isolation [2, 6]. However, these problems in practice can occur simultaneously in time and space. Our key contribution is providing a general diagnostic framework that distinguishes between these problems so that the correct troubleshooting operation(s) can be performed.

4.4. Fault isolation

After residual generation, fuzzy logic based fault detection mechanism fed with the residuals is implemented to determine the existence, the size, and the source of faults in the plant. The purpose is not only to determine whether a fault is present in a system (fault detection), but also the determination of the kind and location (fault isolation), or the determination of the size and time-variant behaviour (fault identification) of the fault [13]. The fault isolation is to isolate the specific component by the ability to re-configure and to process around the bad component. Because of the communication is one of the most important resources in the sensor networks, therefore, our fault isolation scheme providing a testimony-based protocol to isolate a detected node. This testimony-based protocol will apply nodes’ data exchange protocol such as ftp to exchange data between two nodes. Once the healthy nodes have been derived by our fuzzy logic based detection scheme, the healthy node will be aware its neighbouring faulty node in this area network. Then the healthy node will send a Request Reply (RR) packet which contains a simple numeric value. If the faulty nodes can receiving this RR packet and immediately reply the same value back to healthy node, this healthy node will broadcast to their
neighbouring nodes to verify this faulty node still can execute relay node duty. Otherwise, it will broadcast isolating message to their neighbouring nodes to isolate this faulty node. An isolating a faulty nodes means:
(i) Do not route packets through it, to avoid losing them.
(ii) Do not forward packets for it, to insulate it.

4.5. Fault tolerance

To solve the BFP, the cluster heads always can play an important role that use an appropriately protocol to avoid faulty node delivers inaccurate data and assign a health node to replace faulty node. Therefore, to achieve data accuracy, we suggest a consensus based fault tolerance algorithm to overcome Byzantine faults in WSNs. This consensus based algorithm comes from consensus theory [12]. This consensus theory involves general procedures, which summarize estimates from multiple experts decisions based on a Bayesian decision theory assumption. This theory has a combination formula obtained by the consensus rules. Several consensus rules have been proposed. Probably the most commonly used consensus rule is the Linear Opinion Pool (LOP) which has the following (group probability) form for the user specified information (land cover) class \( W_j \) if \( n \) data sources are used:

\[
C_j(Z) = \sum_{i=1}^{n} \lambda_i p(\omega_i | x_i)
\]

Where \( C_j \) is consensus rules, \( j \) is indicate information classes, \( Z = [Z1, Z2, ..., Zn] \) is an input vector, \( p(\omega_i | x_i) \) is a source-specific posterior probability and \( \lambda_i \)'s ( \( i=1,2,\ldots,n \)) are source-specific weights.

The main contribution of this second tier fault tolerance algorithm is to replace the \( \lambda_i \) by the \( \kappa \) parameters. The healthy parameter \( \kappa \) is to express quantitatively the goodness of source data. To clarify our algorithm we provide the following example with the diagrams shown in Fig. 4 (a), (b) to show how it is working with our proposed routing protocol:

(i) After deployment the network will start to organize as a virtual grid as described before. Neighbouring CHs will be organized as a small grid for consensus checking. Each one of these nodes will be the initiator for a certain period of time and then its role changes periodically. The first node can be chosen according to its ID so the first initiator is the node with the lowest ID then the next one, and so on. The initiator is responsible to challenge the other eight neighbouring nodes to collect their health statuses. This is shown in Fig. 4(a) where CH with ID = 5 becomes an initiator.

(ii) The initiator CH will generate challenge data and broadcast it to the vicinity grid nodes. The neighbouring nodes, upon receiving this challenge message, will execute consensus-checking, which uses the challenge list as an input to a computational checking algorithm that performs a series of checks to generate an output message. This checking list can have hardware and / or software checking items. These checking items can be designed according to user requirements.
(iii) Each vicinity grid CH node performs a consensus-checking to assess if any of the nodes returned a result that differs from the expected result. After these grid nodes complete this consensus-checking, they will respond back to the initiator by sending a response message. The initiator will aggregate the response messages and register the node health status results in its memory.

(iv) Based upon the results of this test, the initiator can select healthy grid nodes accordingly. The results can be quantified as parameter $\kappa$ for each CH. The higher value means better healthy condition and vice versa. Once the healthy grid nodes have been listed, the initiator can run the consensus-based decision scheme using equation (2) after replacing $\lambda_i$ for each grid head by its $\kappa$, then the LOP will be executed to get the results $C_i(Z)$. The initiator will send back these results to each grid node. In this algorithm, each node will have a threshold value $T$, this $T$ can be defined by user or sensor application. If the grid node’s result $\lambda_k$ is higher than $T$, it will have high priority to forward the data or event through the network. On the other hand, if the result is lower than $T$, it will become a standby node or only assist in message forwarding.

(v) To maintain this small region grid network operation, our consensus-based algorithm also has a simple replacement scheme to reselect a new grid node. When a grid node fails or becomes standby node, its neighbouring grid node will be aware of that in the short term, thus it will broadcast grid construction message to the area. The first CH node replies to this message will become the new grid node to replace the failed one.

This FTFK consensus-based algorithm diagnoses faulty communication between sensor nodes providing fault detection and isolation to eliminate faulty nodes which affect the network operation and maintenance to correspond to user desired communication quality in the WSNs.

5. Evaluation

5.1. Evaluation setup

We evaluated FTFK using extensive simulation on the Georgia Tech Network Simulator (GTNetS) [11]. GTNetS has provided sensor network module which is a scalable simulation tool designed specifically to support large scale sensor networks simulations. The design of the simulator closely matches the design of real network protocol stacks and hardware. This simulator applies C++ as its programming language and has an object-oriented design, which eases extensibility of existing simulation models. Here, we describe the simulation setup and the metrics examined for the performance evaluation. In order to get the average behaviour, we implemented FTFK in the GTNetS simulator sensor network module, our simulation consists of $(450 \leq N \leq 500)$ sensor nodes including cluster heads in a $1200m \times 1200m$ grid. The networks consist of $(45 \leq N < 50)$ cluster heads which any of them could be a mole node. Fig. 5 shows configured simulation environment.

The simulation time is 200 seconds. Each node has a radio range of 150m. We use few sources and one mobile sink randomly selected from 450 nodes in our simulation. The source sends data every 10 seconds, and the query is periodically sent every 30 seconds. The energy consumptions of transmitting and receiving are 0.66W and 0.395W. To evaluate the performance of FTFK, we compare it to ODDD scheme. We use three main metrics to evaluate the performance of FTFK; namely, Energy Consumption, Transmit Successful Rate and Average End-to-End delay. Due to GTNetS has default one sink limitation, hence we utilized one sink to compare with their scheme. Energy consumption includes that of moles node competition, data dissemination, and the sink mobility.
5.2. Fault injection

For detecting the malfunction cluster heads, we inject interfere packets in the wireless link and lower down the receiving power which attempt to disable packet receiving capability of these cluster heads. Once the fault injection has been done in couple of cluster heads in the network, we assume these faulty cluster heads are not able to receive packet over the network that lower down the packet success ratio and increase end-to-end delay. Then we implement our FTFK fault tolerance mechanism in a schedule to recover the malfunction of the cluster heads. An example of pseudo code as shown in Fig.6, this event-driven FTFK mechanism can leads to regularly execute segments of code to provide energy efficiency within a while() block.

5.3. Average success ratio

This experiment is mainly to measure average success ratio under a number of cluster heads failure in the networks. The average success ratio is ratio of successfully packets deliver to mobile sink. Hence, in this experiment, we set up mobile sink move at different speed (0, 5, 10, 15, 20 m/sec) and a number of cluster heads failure from 0 to 12 (0, 3, 6, 9, 12 nodes). As the diagram shows in Fig.7 (a), the cluster heads failure has impact against on the average success ratio although mobile sink can collect data from a sender by move around in the network. The average success ratio has dramatically decreased when failed cluster heads increase.
On the other hand, as shown in Fig. 7 (b), we observe that FTFK mechanism is able to maintain high success ratios above 82%. Once the node has been recovered from the faulty state, the average packet success ratio will be increased to a higher value comparing to the non-maintenance situation.

5.4. Average energy dissipation

In the following experiment, we investigate the average energy dissipation after the failure on each cluster head has been recovered. We set the number of failure cluster heads are varied from 0 to 12 and one sink move by different speeds (0, 20, 40, 60 m/sec). This experiment is to measure the average energy dissipated on each cluster heads in the networks. Fig.8 shows that average energy consumption on each cluster heads without FTFK mechanism implemented. The node energy consumption has little decreased compared to Fig.8. As exhibited in Fig.8, the higher energy dissipation is due to each cluster head executes fault examine and reconfigure tasks in the WSNs.

Since number of failure cluster heads are varied from 0 to 12 and reputation checker will check periodically for every 50 seconds (50, 100, 150, 200, 250 seconds). A number of cluster heads will have failure(s) occurred during simulation perform. Once CMM mechanism starts on its execution at examine time points, the CMM mechanism will be continued until the end of recovery procedure. The duration to recover failure occurs is examine time for our CMM mechanism. As exhibited in Fig.9, examine time is between 0.0381 and 0.0406 seconds. The reason for this time variation may depend on node processing speed and complexity of the implementation of CMM mechanism. In this experiment, we understand that the complexity of CMM mechanism will influence the node examine time.
6. Conclusion & Future Works

In this paper, we introduce a new fault model for arbitrarily node behaviour in heterogeneous WSNs. We design a fault-tolerance algorithm based on the cluster-based framework and show how this model works to find out the probably fault nodes. Our algorithm also adapts the Bayesian approach to observe and estimate nodes reputation regarding message forwarding. The simulation results show that our FTFK scheme expense little bit higher energy and examine time to achieve better average end to end delay and packet success ratio when a number of cluster heads failure in the WSNs. For our future work, we will dedicate to improve its power efficiency issue to adapt future application requirements.

7. References
