Fault diagnosis in mobile computing using TwinSVM

Diagnóstico de fallas en computación móvil usando TwinSVM

Diagnóstico de falhas em computação móvel usando TwinSVM

Neha Malhotra[1]
Manju Bala[2]
Vikram Puri[3]

Received: January 10th, 2022
Accepted: April 15th, 2022
Available: April 30th, 2022

How to cite this article:
N. Malhotra, M. Bala and V. Puri, "Fault diagnosis in mobile computing using TwinSVM," *Revista Ingeniería Solidaria,* vol. 18, no. 1, 2022. doi: https://doi.org/10.16925/2357-6014.2022.01.11

Research article. https://doi.org/10.16925/2357-6014.2022.01.11

[1] Researcher, I. K. Gujral Punjab Technical University and Assistant Professor, Lovely Professional University, India
Email: neha.16982@lpu.co.in
ORCID: https://orcid.org/0000-0002-7850-3658

[2] Director, Khalsa College of Engineering & Technology, India.
Email: drmanjubhp@gmail.com
ORCID: https://orcid.org/0000-0002-2752-685X

[3] Researcher, Duy Tan University
Email: purivikram@duytan.edu.vn
ORCID: https://orcid.org/0000-0002-4023-9935

Abstract

Introduction: this paper is the outcome of the research "fault diagnosis in mobile computing using TwinSVM" developed at I.K Gujral Punjab Technical University in Punjab, India in 2021.

Problem: since resources in Mobile Computing Systems are limited, and a system has limited bandwidth, energy, and node mobility, the desired network behaviour may change if faults are present.

Objective: to achieve fault tolerance, so that a mobile system can operate even in the presence of faults, a two-timer approach was implemented in the detection framework, which was then enhanced and refined with the use of the TwinSVM classifier. This classifier aids in identifying outlier nodes, making the approach more fault tolerant.

Methodology: the monitoring framework classifies the detected node as normal, faulty, or partially faulty, by initiating a heartbeat check timer and another relevance check timer in case the node doesn't respond to the first timer, which is further tested using TwinSVM, which improves its efficiency by detecting outliers.

Results: the proposed framework performs better in terms of detection accuracy, energy consumption, latency, and packet drop ratio, all of which have been improved.

Conclusion: Fault diagnosis using the TwinSVM machine learning classifier performs better in terms of false alarms and false positive rates and is suitable for providing fault tolerance in mobile computing systems.

Originality: through this research, a unique version of fault detection in mobile computing has been developed using a classifier-based approach.

Limitations: the lack of other fault detection techniques falls under fault classification.

Keywords: Faults, Fault Tolerance, Mobile Computing, Fault classification, Fault Diagnosis, TwinSVM.

Resumen

Introducción: este artículo es el resultado de la investigación "Diagnóstico de fallas en la computación móvil usando TwinSVM" desarrollada en la Universidad Técnica I.K Gujral Punjab en Punjab, India en 2021.

Problema: dado que los recursos en los sistemas informáticos móviles son limitados y un sistema tiene un ancho de banda, energía y movilidad de nodos limitados, el comportamiento deseado de la red puede cambiar si hay fallas.

Objetivo: para lograr la tolerancia a fallas, de modo que un sistema móvil pueda operar incluso en presencia de fallas, se implementó un enfoque de dos temporizadores en el marco de detección, que luego se mejoró y perfeccionó con el uso del clasificador TwinSVM. Este clasificador ayuda a identificar nodos atípicos, lo que hace que el enfoque sea más tolerante a fallas.

Metodología: el marco de monitoreo clasifica el nodo detectado como normal, defectuoso o parcialmente defectuoso, iniciando un temporizador de verificación de latidos y otro temporizador de verificación de relevancia en caso de que el nodo no responda al primer temporizador, que se prueba más usando TwinSVM, que mejora su eficiencia mediante la detección de valores atípicos.

Resultados: el marco propuesto funciona mejor en términos de precisión de detección, consumo de energía, latencia y relación de caída de paquetes, todos los cuales han sido mejorados.

Conclusión: el diagnóstico de fallas que utiliza el clasificador de aprendizaje automático TwinSVM funciona mejor en términos de falsas alarmas y tasas de falsos positivos y es adecuado para proporcionar tolerancia a fallas en sistemas informáticos móviles.

Originalidad: a través de esta investigación, se ha desarrollado una versión única de detección de fallas en computación móvil utilizando un enfoque basado en clasificadores.

Limitaciones: la falta de otras técnicas de detección de fallas cae dentro de la clasificación de fallas.

Palabras clave: fallas, tolerancia a fallas, computación móvil, clasificación de fallas, diagnóstico de fallas, TwinSVM.

Resumo

Introdução: Este artigo é o resultado da pesquisa "diagnóstico de falhas na computação móvel usando TwinSVM" desenvolvida na I.K Gujral Punjab Technical University em Punjab, Índia em 2021.

Problema: Como os recursos em sistemas de computação móvel são limitados e um sistema tem largura de banda, energia e mobilidade de nó limitados, o comportamento de rede desejado pode mudar se houver falhas.

Objetivo: Para alcançar a tolerância a falhas, para que um sistema móvel possa operar mesmo na presença de falhas, uma abordagem de dois temporizadores foi implementada na estrutura de detecção, que foi aprimorada e refinada com o uso do classificador TwinSVM. Esse classificador auxilia na identificação de nós discrepantes, tornando a abordagem mais tolerante a falhas.

Metodologia: A estrutura de monitoramento classifica o nó detectado como normal, defeituoso ou parcialmente defeituoso, iniciando um cronômetro de verificação de pulsação e outro cronômetro de verificação de relevância caso o nó não responda ao primeiro cronômetro, que é posteriormente testado usando TwinSVM, que melhora sua eficiência detectando valores discrepantes.

Resultados: O framework proposto apresenta melhor desempenho em termos de precisão de detecção, consumo de energia, latência e taxa de queda de pacotes, todos eles aprimorados.

Conclusão: O diagnóstico de falhas usando o classificador de aprendizado de máquina TwinSVM apresenta melhor desempenho em termos de falsos alarmes e taxas de falsos positivos e é adequado para fornecer tolerância a falhas em sistemas de computação móvel.

Originalidade: Através desta pesquisa, uma versão exclusiva de detecção de falhas em computação móvel foi desenvolvida usando uma abordagem baseada em classificadores.

Limitações: A falta de outras técnicas de detecção de falhas se enquadra na classificação de falhas.

Palavras-chave: Falhas, Tolerância a Falhas, Computação Móvel, Classificação de Falhas, Diagnóstico de Falhas, TwinSVM.

1. INTRODUCTION

The adoption of mobile computing systems in various fields has matured tremendously. The main reason is the high-level performance of such systems in mobile environments. Mobile computing systems are a set of processes that communicate with each other using wireless networks. In these networks, all the nodes are called as Mobile Hosts (MH) and communicate with each other using Mobile Support Stations (MSS). Faults can be triggered by the malfunctioning of any of the nodes in the system or the transmission of false information by the neighbouring nodes to all the nodes in the network [1].

Fault management in mobile computing requires three major steps: fault detection, positioning of faulty nodes and node recovery [2]. Machine learning approaches help to automate and refine the system. With the introduction of efficient

fault classifiers like "TwinSVM", fault classification has been improved resulting in increased detection accuracy and a reduction in false alarm.

One of the most popular methods is the comparison-based approach, proposed by [3][4], where all the nodes participating in the test determine whether the neighboring nodes are faulty or fault free. The existing approaches fall into the category of centralized, distributed or hybrid approaches where fault detection is the core responsibility of the central node, a set of nodes or a combination of both respectively [5]. The research community has faced different faults which can be categorized and described below [6]:

Out of bound fault: When readings of the nodes are out of bound from the normal/ desired readings.
Stuck at fault: When there is no variation in the readings i.e., it is sensed to be zero.
Gain Fault: When the variation is high compared to the predefined threshold.
Offset Fault: When the variance is low compared to the predefined threshold.

The proposed fault tolerance framework uses a hybrid approach in which fault detection is the core responsibility of all the nodes in the network along with the base station. In this system, the network has been continuously monitored to observes the deviation from the normal results and, in case there is a deviation in the results found, a fault detection algorithm using machine learning is executed to detect the faulty node in the system which refines the existing system to classify the faults after its detection. The proposed structure aims to provide the system with effective fault detection accuracy, efficient detection latency, minimum energy consumption etc.

1.1. Literature Review

The author [7] has presented a distributed fault detection approach where each node collects information from their neighboring nodes to avoid communication overhead and then a majority voting-based scheme has been followed to elect the faulty node in the network. The author [8] has proposed a supervised machine learning based approach to detect the faults in the system in which predefined sets which have been generated from fault occurrences in the network using a randomized extra trees approach. The authors [9] have introduced a time out mechanism, analysis of variance and probabilistic neural network approaches to detect hard, soft and type of fault in the network in three phases of clustering, detection and fault classification.

The authors [10] have developed an automated fault diagnosis framework to achieve fault tolerance in different phases like initialization, classification and tolerance of faults. As per the methodology, the hard and link faults have been diagnosed using a checksum method whereas soft faults have been diagnosed using statistical tests. The said technique is helpful in the development of fault detection frameworks. The author [8] has proposed a tree-based fault diagnosis for the detection of faults in a network. The said technique has evolved to consume low training time in comparison to others. The authors [11] have introduced a probabilistic outlier detection scheme to detect the outliers in the wireless sensor network based on copula theory. The performance of this technique is better than other statistical tests for outlier detection. The authors [12] have presented a distributed fault detection and diagnosis framework in which the faults have been initially detected by a sensor device and then by the central node; the result of which is the instant detection of faults while reducing computational overhead. The author [13] has detailed a scheme based on varying time topologies; a distributed fault diagnosis scheme focuses on the decomposition schemes and its deterioration and pre training problems. The authors [14] have presented a distributed fault detection framework in which a credibility model has been established to identify the status of nodes; transmission overheads have been reduced by limiting the number of diagnosing requests. The author [15] identified that performance of the fault detection method has been analyzed using different classifiers so that efficiency of the proposed scheme can be easily identified. The authors [16] have defined petri nets using a trust based model for the detection of faults in the network where TFM has been used for the building of compound model structures.

2. PROPOSED FAULT DIAGNOSIS FRAMEWORK

An optimized fault diagnosis framework for mobile computing systems has been proposed to efficiently monitor the MCS to detect the faults and also does classification of different types of faults using a classifier. In this section, a system model of the proposed framework has been introduced which includes the algorithm for system diagnosis and the detection of faults and an algorithm for the classification of faults.

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

2.1. System Model

2.1.1. Assumptions:

a) All the processes/nodes in the mobile computing environment are uniformly distributed.
b) Each process/node has been identified by its unique id which is static in nature.
c) All the processes/nodes are assumed to have uniform structure in the form of transmission range and energy consumption.
d) All the processes/nodes send their data in a multi hop fashion.
e) The partition managers are elected based on the number of nodes in the system.
f) Based on the time of their arrival in the network, the nodes with a higher energy level will be elected as partition managers.

2.1.2. Network Model

All the nodes have been randomly deployed and nodes with a higher energy level, more power and transmission range have been elected as partition managers; which form up the cluster in the network as depicted in Figure 1, which represents the network in a two-dimensional plane. Partition managers will broadcast in the system and all the nodes in its range calculate the received signal strength indicator (RSSI) after receiving the signal from its partition manager, which is measured based upon the power present in the signal. Based on the power level and its design, the transmission power level is to be set as 0 dBM and receiving sensitivity range. The RSSI for the nodes present in the system can be calculated using the Equation (1):

$$\boldsymbol{x_{pr} = x_{pt} - x_{pl}(d_i) \times \mu \, log_{10}\left(\frac{d}{d_i}\right)} \tag{1}$$

Where x_{pr} is the RSSI, $x_{pt}$is the transmitting power, x_{pl} is the reference power level, d_i is the reference distance, d is the distance between nodes and μ is the space coefficient which is defined in Equation (2):

$$\boldsymbol{\mu = \begin{cases} 2, d_i \leq d \\ 3, d_i > d \end{cases}} \tag{2}$$

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

As defined in the equations above, the node becomes a partition manager of the partition for which all the nodes with RSSI value lie within the minimum threshold range. Each node of the partition communicates with each other through the partition manager and the partition managers communicate with each other through the base station.

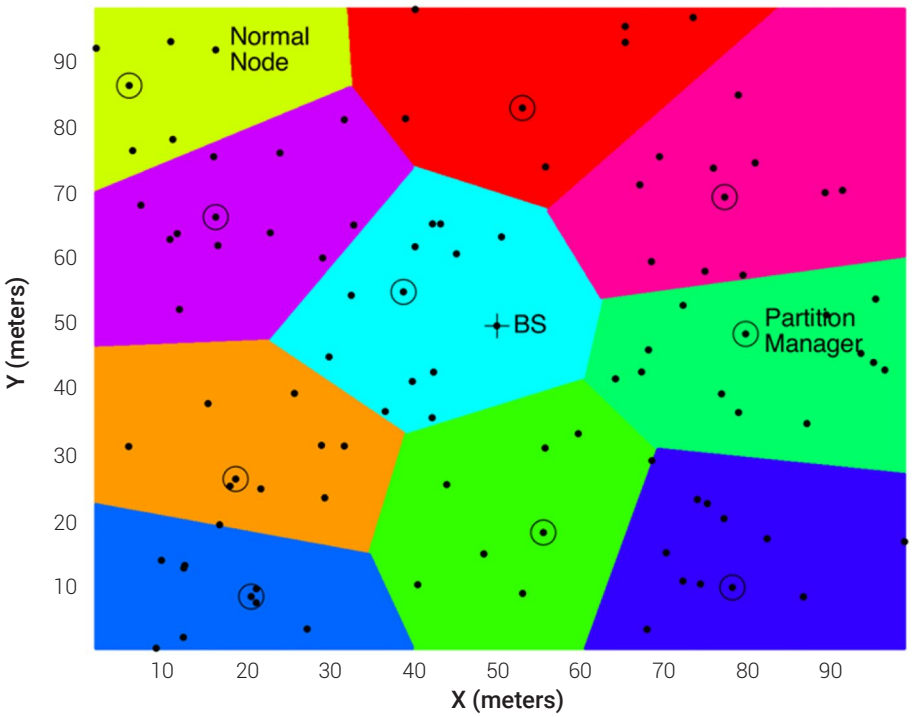

Figure 1. Network Model showing Normal nodes managed by the partition manager and the base station
Source: own work

2.2. Network Monitoring Framework

Monitoring and detection (Node level): Partition managers initiate the monitoring of the partition and then send the results to the base station. Different timers have been proposed to achieve fault tolerance in the system and are explained below:

T(Tu) is used to set the frequency at which the node sends the status to the partition manager node. **T(Tm)** sets the monitoring time interval for the partition manager to check the status of the nodes. **T(Tint)** is the grace timer which gets activated once the response from the neighboring node has not been received by the time Tm ends. Before declaring the node as faulty, the master node initiates a grace timer (Tint) which is also a second timer to confirm if the targeted node is faulty or not.

If a node(m) is not getting heartbeats from the next neighboring node (m+1), it waits for T(m) and then checks with the immediate next neighbor node (m+2) of m+1 for the status of (m+1) by sending an update message.

The partition manager will initiate a health check in the partition by sending an update packet to all the nodes in the partition and initiates Tm and the nodes responds with a response packet to mark their presence in the network. The RESP packet consists of the coordinates of nodes, energy levels, sequence numbers and IDs). If a node replies with a RESP packet, it is considered as possible fault_free and will go for reliability_check(). Elseif, a node 'm' does not respond with RESP packet and Tm expires, then the partition manager will initiate interim period T(int) to confirm the status of node 'm'. If it doesn't receive the heart_beat and if the second timer expires, it will broadcast to the partition manager about the breakout of m+1. The step-by-step approach has been detailed in Figure 2.

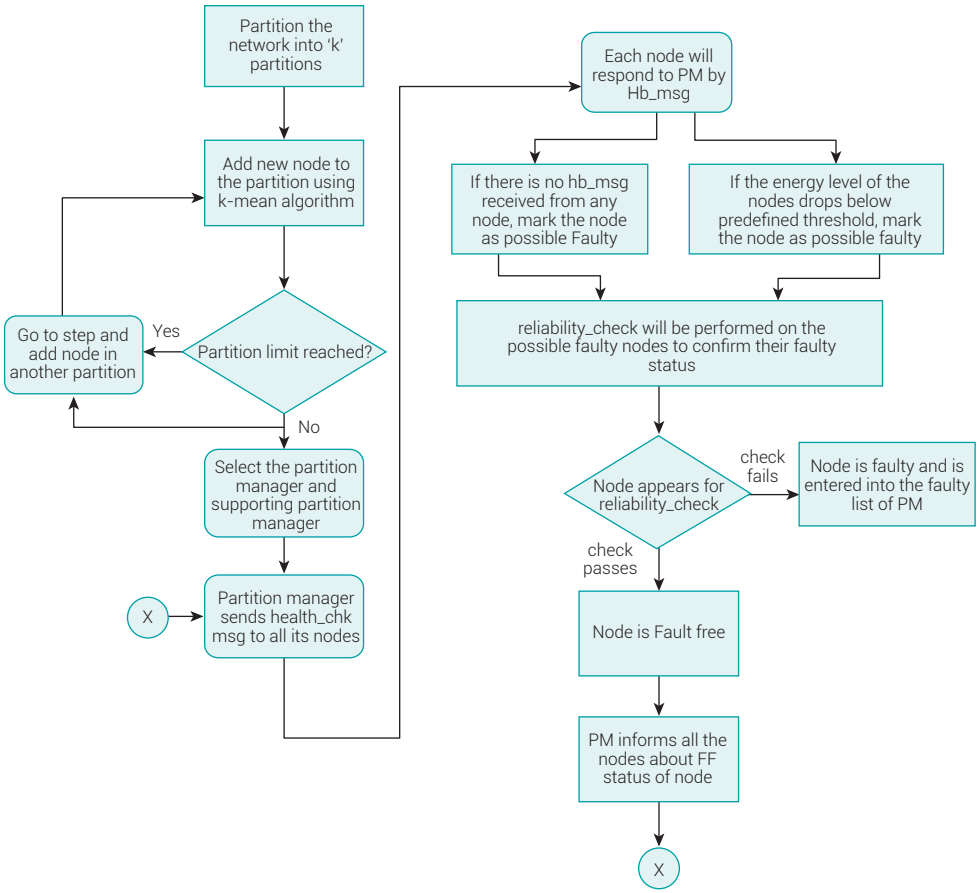

Figure 2. Fault detection framework
Source: own work

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

Algorithm 1: A Distributed neighborhood comparison based fault diagnosis framework

Input: Collected Statistics $\boldsymbol{S}$ from the nodes $\boldsymbol{x} \in \boldsymbol{N}$, where 1≤i ≤ r, r=number of rounds and i = time instance. Initialize T(m) = x; T(int) = y; set $\boldsymbol{N}$ [$\boldsymbol{class}$] = $\boldsymbol{normal}$, $\boldsymbol{N} \in \mathbf{N}$, (1 ≤ j ≤ $\boldsymbol{N}$) default class of node is initialized to be normal

1. *for* each node $\boldsymbol{N} \in$ N, (1 ≤ j ≤ $\boldsymbol{N}$) do
2. S[HEALTH] = NODE_HEALTH_CHECK(N_j), *Delay* N_j[T(m)] until x
3. *If* S[HEALTH] is not empty
4. S[RELIABILITY_CHECK] = check_invalidation_rule(N_j) *Else*
5. $N_{neighbor\_node}$ = Euclidean_distance(N_j)
6. S[N_HEALTH] = NODE_HEALTH_CHECK_NEIGHBOR($N_{neighbor\_node}$, N_j)
7. *Delay* N_j[T(int)] until y
8. *If* S[N_HEALTH] is empty()
9. N[class] = faulty
10. BROADCAST_TO_PARTITION_MANAGER(N[class])
11. *end*
12. *If* S[N_HEALTH][Last_Received] < S[HEALTH][Last_Received]
13. N[class] = tentative_faulty
14. BROADCAST_TO_PARTITION_MANAGER(N[class]) *Else*
15. *Repeat the testing of the node* N_j
16. *End*
17. *End*
18. *END*

Once a node is marked as evicted, two possible cases arise thereafter:

Case 1: The node joins the same partition again: Since the mobile node consumes energy during its activity, leading to the depletion of its energy, when it restores its energy level and tries to connect to the same partition from which it came, the access point on connection request broadcasts it to the partition manager to restore the node. The partition manager will move the node from the faulty list to the active list.

Case 2: The node moved to another partition/location /access point: In this case, the access point to which the node is now connected, broadcasts to other access points about its connection and if any of the other access points have the information about that node, it will inform its own partition manager to remove the node (m+1) as the node(m+1) becomes the part of a new partition.

The invalidation rule, as described by authors in [17] and depicted in Table 1, has been used to perform the reliability_check of the nodes. These invalidation rules have been defined in literature and are used to validate the decision of marking the node as faulty/normal.

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

Table 1. Invalidation rules used for node reliability checks

Status is 0 if sequence id of the node is incremented else it is 1	Status is 0 if energy level decreases from predefined value else it is 1	Reliability_check() output, 0 means node is fault free and 1 means node is faulty
0	0	0
1	0	1
0	1	1
1	1	1

2.3. Fault Classification

The main reason to build a classifier is for the improvement of classification results which eventually helps to achieve and improved the belief function [15]. The fault classification constitutes two parts; one being the Fault Classification algorithm, and the other is the TwinSVM classifier model. In TwinSVM both linear and non-linear models have been described. The fault classification algorithm is described as follows.

2.3.1. Fault Classification algorithm

The job of the fault classification algorithm is to provide a set of examples to the TwinSVM model as defined in the next section. The fault classification algorithm works by assigning classes to the collected samples out of the monitoring framework. The collected statistics S from the sensor nodes are assigned a class [*normal, partial fault, damaged*] utilizing heartbeat, energy and sync delay as defined in the algorithm. θ, φ and δ are the predefined threshold values of good sensor nodes for energy, PDR and Latency, using these ranges we can assign the class to the respective instance.

Algorithm 2: Fault classification algorithm

Input: Collected Statistics $\boldsymbol{S}$ from the Sensor nodes $\boldsymbol{x}_i \in \boldsymbol{N}$, where 1≤ i ≤ r, r=number of rounds and i = time instance. Initialize: $\boldsymbol{\theta}$, $\boldsymbol{\varphi}$ and $\boldsymbol{\delta}$, θ are the predefined range or threshold values of good sensor nodes for energy, PDR and latency. Set $N_j[\boldsymbol{class}] = \boldsymbol{normal}$, $\boldsymbol{N_j \in N}$, (1 ≤ j ≤ $\boldsymbol{N}$) default class of node is initialized to be normal,

1. ***for*** each node $\boldsymbol{N_j} \in$ N, (1 ≤ j ≤ $\boldsymbol{N}$) do
2. ***if*** *S[Heartbeat]* = = 0 and *S[HeartbeatN]* = = 0
3. *N*[*class*] = faulty, mark node faulty
4. ***End***
5. ***if*** *S[Heartbeat]* = = 0 and *S[HeartbeatN]* = = 1
6. *N*[*class*] = *faulty,*
7. ***End***
8. ***End***
9. ***for*** each node $\boldsymbol{N_j} \in$ N, (1 ≤ j ≤ $\boldsymbol{N}$) do

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

10. **if** N_j[*class*] == *faulty,*
11. **if** *S[PDR]* > φ or *S*[*Energy*] < θ
12. *N*[*class*] = *damaged,*
13. ***End***
14. **if** *S[PDR]* > φ and *S*[*Energy*] > θ
15. *N*[*class*] = *partial fault*
16. ***End***
17. **if** (*S*[$SYNC_{DELAY}$] > δ) and (*S*[*Energy*] > θ and *S*[*PDR*] < φ)
18. *N*[*class*] = *partial fault*
19. ***End***
20. ***End***
21. ***End***

The stated approach works in the following phases:

a) The observed readings will be used as input to the algorithm.
b) In Step2, faults like gain fault, offset fault, stuck at fault and out of bound faults will be induced into the system.
c) In Step3, machine learning based classifiers will be used for classification, i.e. BPNN, GA and TwinSVM. The said techniques formulate a decision function based on the observations. Once the decision function is deployed, output data will be classified into three classes i.e., normal class, partial faulty class and faulty class.

2.4. Classifier Used

The basic purpose of classifiers is to classify an observed value into different predefined categories/classes by taking data sets as inputs which contain the same observations [15]. Different classifiers have been introduced in the literature for fault detection and few have been selected to be implemented on our detection framework which narrowed down our search to use the following classifiers for our fault detection framework.

2.4.1. Twin Support Vector Machine Classifier

Support Vector Machine (SVM) [18] is a classification and regression algorithm which can be used to solve linear and non-linear problems and works well when a large number of data samples are available. The SVM works by creating a hyperplane which is used to separate the data instances into classes. The SVM has two models based on linear and non-linear kernels.

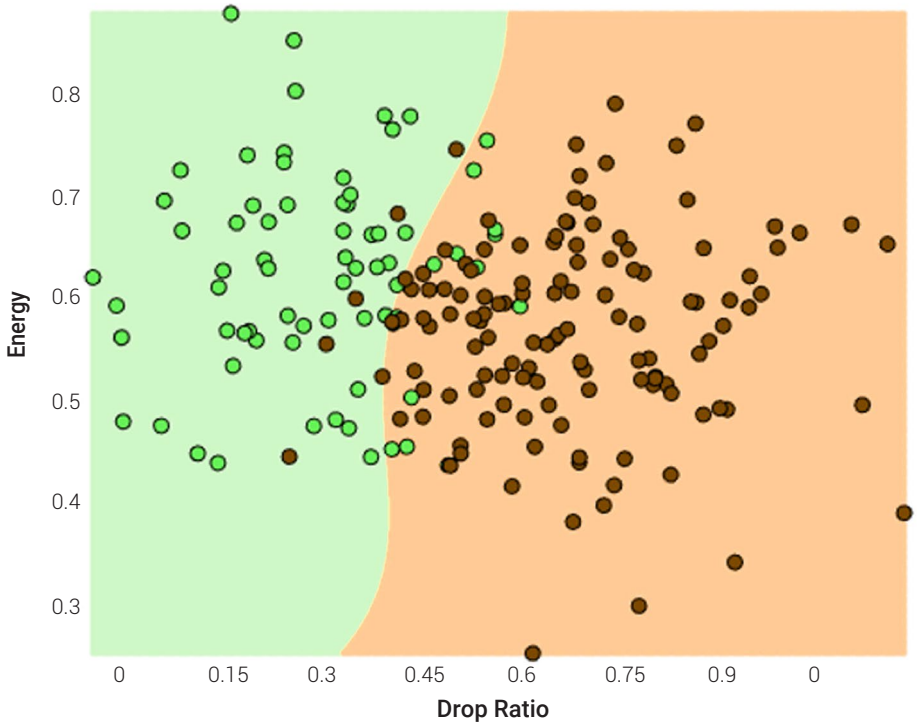

Figure 3. Decision boundary for the TwinSVM classifier
Source: own work

The decision to use a linear or nonlinear kernel relies on two main factors; one being the performance in terms of speed and the other being predictive performance i.e., accuracy, as node classification for the linear kernel is much faster. However, the best possible predictive performance is better when using a nonlinear kernel, as the linear kernel is less accurate than a properly tuned non-linear kernel. For the fault classification Twin support vector machine (TwinSVM)][20] is used. TwinSVM is a machine learning approach which can be used for the classification of the faults or classification of faulty as well as normal nodes in the network. TwinSVM was proposed by [20] to achieve the best of both worlds. TwinSVM is not only fast, but also shows good predictive performance. Figure 3 above shows the two-dimensional scatter plots of the collected node data where node energy and the Packet Drop Ratio (PDR) is used as a deciding factor for classification in the work.

Data for training of the TwinSVM in generated using the classifier algorithm previously mentioned, which is collected during simulation runs. We assume that a faulty set of nodes is represented by class 'c_f' and a normal set of nodes are represented by class 'c_n' respectively. The data samples of these classes are represented using the matrices in Equations (3) & (4)

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

$$Z_1 \in T^{cf*k} \ and \ Z_2 \in T^{cn*k} \tag{3}$$

where 'T' represents a k-dimensional space. For the given k-dimensional plane, equations have been given below [21]:

$$x^T u_1 + s_1 = 0 \ and \ x^T u_2 + s_2 = 0 \tag{4}$$

Where, u_1 and u_2 are the normal vectors in the plane and s_1 and s_2 are given bias terminologies. To build up the TwinSVM classifier, Equations (5) and (6) will be used, where the symbols ξ and ζ are the slack variables, d_1 and d_2 are penalty parameters, and n_1 and n_2 are suitable dimension vectors with values = 1.

$$\min\left(u_1, s_1, \xi\right) \frac{1}{2} \|Z_1 u_1 + n_1 s_1\|^2 + d_1 n_2^T \xi$$
$$\text{s.t.} -(Z_2 u_1 + n_2 s_1) + \xi \geq n_2, \xi \geq 0 \tag{5}$$

$$\min\left(u_2, s_2, \zeta\right) \frac{1}{2} \|Z_2 u_2 + n_2 s_2\|^2 + d_2 n_1^T \zeta$$
$$\text{s.t.} (Z_1 u_2 + n_1 s_2) + \zeta \geq n_1, \zeta \geq 0 \tag{6}$$

Equation (7) represents the lagrangian of Equation (5):

$$\mathcal{L}\left(u_1, s_1, \xi, \gamma, \delta\right) = \frac{1}{2} \|Z_1 u_1 + n_1 s_1\|^2 + d_1 n_2^T \xi + \gamma^T ((Z_2 u_1 + n_2 s_1) - \xi + n_2) \delta^T \xi \tag{7}$$

γ, δ are the two lagrangian multipliers in (5)
Few merging may result in (8):

$$\begin{bmatrix} Z_1^T \\ n_1^T \end{bmatrix} [Z_1 \quad n_1] \begin{bmatrix} u_1 \\ s_1 \end{bmatrix} + \begin{bmatrix} Z_2^T \\ n_2^T \end{bmatrix} \gamma = 0 \tag{8}$$

By assuming, $M = [Z_1 \quad n_1]$ and $N = [Z_2 \quad n_2]$ $a_1 = \begin{bmatrix} u_1 \\ s_1 \end{bmatrix}$, formulate

$$M^T M a_1 + N^T \gamma = 0 \tag{9}$$

$$a_1 = -(M^T M)^{-1} N^T \gamma \tag{10}$$

When it is challenging to obtain the inverse of $M^T M$, a regularisation term '$\delta 1$' can be added, where 'I' is an identity matrix which reformulates Equation (10) [19] as Equation (11):

$$a_1 = -(M^T M + \delta I)^{-1} N^T \gamma \tag{11}$$

Normal vector and bias class for 'a_2' can be obtained in (12):

$$a_2 = (N^T N + \delta I)^{-1} M^T \gamma \tag{12}$$

Likely, TwinSVM determines a hyper plane and a new data member can be assigned to a class 'i' with the help of the function mentioned in Equation (13):

$$\text{Class i} = \min |x^T u_i + S_i| \quad for\ i = 1, 2 \ldots \tag{13}$$

For every new node in the system, its perpendicular distance will be calculated from each hyper plane and whichever distance is small, that pattern is assigned. The data samples which cannot be easily classified using Linear TwinSVM, to make clear and easy classification for such data samples, kernel functions may be used [19]. Equations for non-linear TwinSVM can be formulated as in Equation (14):

$$\begin{aligned} min(w_1, s_1, \xi) \quad & = \frac{1}{2} \| P(Z_1, U^T) w_1 + f_1 s_1 \|^2 + d_1 f_2^T \xi \\ \text{s.t.} \quad & -(P(Z_2, U^T) w_1 + f_2 s_1) + \xi \geqslant f_2, \xi \geqslant 0 \end{aligned} \tag{14}$$

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

$$\boldsymbol{min}(\boldsymbol{w_2}, \boldsymbol{s_2}, \boldsymbol{\eta}) \frac{\mathbf{1}}{\mathbf{2}} \|\boldsymbol{P}(\boldsymbol{Z_2}, \boldsymbol{U^T})\boldsymbol{w_2} + \boldsymbol{f_2 s_2}\|^2 + \boldsymbol{d_2 f_1^T \eta} \tag{15}$$

$$\text{s.t.}(\mathrm{P}(\mathrm{Z}_1, \mathrm{U}^{\mathrm{T}})\mathrm{w}_2 + \mathrm{f}_1\mathrm{s}_2) + \eta \geqslant \mathrm{f}_1, \eta \geqslant 0$$

In which U= $[Z_1 \ Z_2]^2$ and the kernel function for the same has been denoted by 'P' as in Equation (16);

$$\boldsymbol{P}(\boldsymbol{x^T}, \boldsymbol{U^T})\boldsymbol{w_1} + \boldsymbol{s_1} = \mathbf{0} \textbf{ and } \boldsymbol{P}(\boldsymbol{x^T}, \boldsymbol{U^T})\boldsymbol{w_2} + \boldsymbol{s_2} = \mathbf{0} \tag{16}$$

Equation (17) represents the lagrangian of Equation (15):

$$\begin{aligned} \boldsymbol{L}(\boldsymbol{w_1}, \boldsymbol{s_1}, \boldsymbol{\xi}, \boldsymbol{\gamma}, \boldsymbol{\delta}) = \frac{\mathbf{1}}{\mathbf{2}} \|\boldsymbol{P}(\boldsymbol{Z_1}, \boldsymbol{U^T})\boldsymbol{w_1} + \boldsymbol{f_1 s_1}\|^2 + \boldsymbol{d_1 f_2^T \xi} \\ + \boldsymbol{\gamma^T}\Big((\boldsymbol{P}(\boldsymbol{Z_2}, \boldsymbol{U^T})\boldsymbol{w_1} + \boldsymbol{f_2 s_1}) - \boldsymbol{\xi} + \boldsymbol{f_2}\Big) - \boldsymbol{\delta^T \xi} \end{aligned} \tag{17}$$

Bias and normal vectors of non-parallel planes can be formulated as in Equations (18), (19) and (20):

$$\boldsymbol{g_1} = \begin{bmatrix} \boldsymbol{w_1} \\ \boldsymbol{s_1} \end{bmatrix} = -(\boldsymbol{R^T R})^{-1} \boldsymbol{H^T \gamma}. \tag{18}$$

$$\boldsymbol{g_2} = \begin{bmatrix} \boldsymbol{w_2} \\ \boldsymbol{s_2} \end{bmatrix} = (\boldsymbol{H^T H})^{-1} \boldsymbol{R^T \gamma} \tag{19}$$

where $\mathrm{R}=[\mathrm{P}(\mathrm{Z}_1, \mathrm{U}^{\mathrm{T}})\ \mathrm{f}_1]$ and $\mathrm{H}=[\mathrm{P}(\mathrm{Z}_2, \mathrm{U}^{\mathrm{T}})\mathrm{f}_2]$ sample is classified as

$$\textbf{Class } \boldsymbol{i} = \boldsymbol{min}|\boldsymbol{P}(\boldsymbol{x^T}, \boldsymbol{U^T})\boldsymbol{w_i} + \boldsymbol{s_i}| \textbf{ for } \boldsymbol{i} = \mathbf{1}, \mathbf{2}. \tag{20}$$

For every new node in the system, its distance is measured from both ends of the Kernel surfaces and a new class from which the distance is less will be assigned to the new node.

3. RESULTS

A simulation of the proposed framework has been made using the Random waypoint mobility model and performance evaluation for the same has been performed on MATLAB 2019a based on the simulation parameters listed in Table 2 in the network depicted in Figure 4, where the base station is located in the middle of the network area:

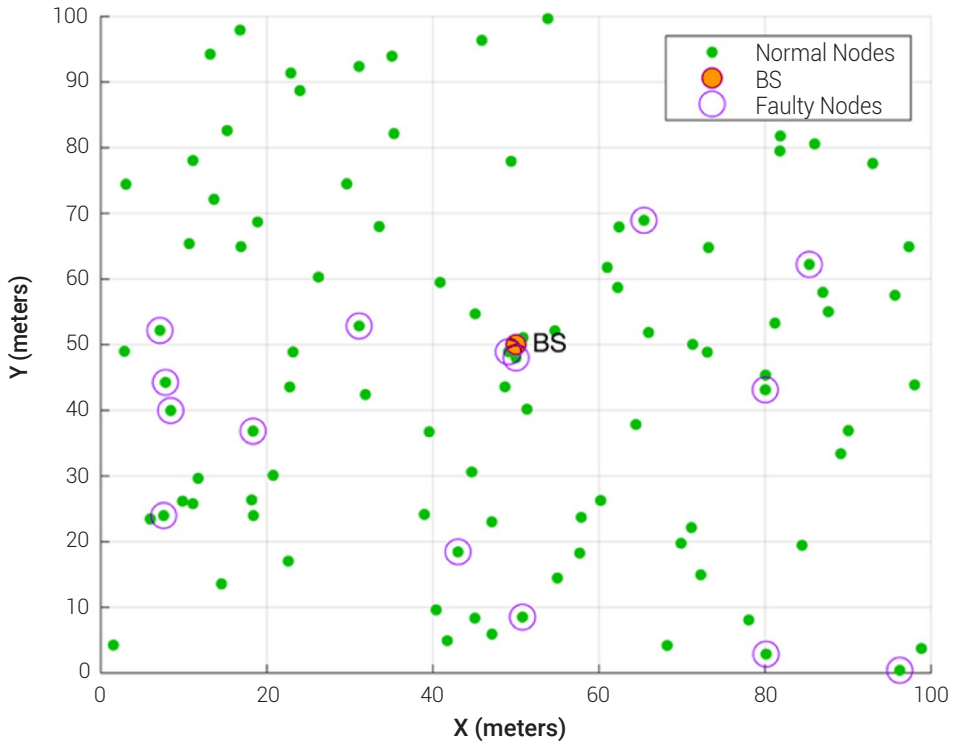

Figure 4. Network with normal and faulty nodes
Source: own work

3.1. Simulation parameters:

The nodes in the network have been deployed in the area of 100*100 m^2. All the simulation parameters used have been presented in Table 2. The sensor nodes usually generate data packets in an acceptable range. But the faulty nodes in the network give results different from the desired result of the acceptable range. This abnormal behavior of faulty nodes is introduced using packet forwarding estimation values.

Table 2. Network Simulation Parameters

Parameter	Value
Energy	10 Joules
Area of deployment	100 x 100
Range of transmission	150 meters
Size of packet	32 bytes
Number of nodes	100-1000
SYNC duration	8.4 ms
Power (receiving)	83.1 mW
Power (idle)	105 nW
Power (transmit)	52.2 mW
Power (sleep)	48 nW
Mobility	Random Waypoint
Fault Classifier	TwinSVM
Kernel	Gaussian RBF
Normalization	Min-Max
Hyper parameters C1	0.1
Hyper parameters C2	0.5
K-Fold Validations	5

If the node is partially faulty, then its packet forward range may vary from 0.05 to 0.07. If a node is normal, then its packet forwarding estimation values may range from 0.07 to 0.10. If a node is faulty, then its packet forwarding estimation will be less than 0.01. Number nodes in the network gradually increases from 100 to 1000 to evaluate the performance at different sets.

3.2. Performance Evaluation

The performance of the proposed model has been analyzed with the existing model and the performance of the fault detection and monitoring framework has also been analyzed on the basis of the parameters listed below:

a) *Network False Alarm Ratio*: The ratio between the number of false alarms (normal nodes identified as faulty) and total number of alarms (total normal nodes) in the network.

b) *Network False Positive Rate*: The ratio between the number of faulty nodes identified as normal and total number of faulty nodes in the network.

c) *Network Detection Latency*: Time taken by the protocol to detect faults in the network

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

d) *Network Detection Accuracy*: The ratio between the number of faulty identified nodes and the actual number of faulty nodes in the network.

e) *Network Energy Consumption*: The total energy consumed in detecting faults in the network.

3.2.1. MSE versus Number of Nodes

Fitness of the proposed solution is calculated by mean square error (MSE) as given in Equation (21) and in Table 3, where Ni is the total number of instances; e.g., if there are 1000 nodes in the network, Ni will be 1000. The target is the target class (Normal, Partial Faulty, Damaged) of the current instance and output is the computed output.

$$\boldsymbol{MSE = \frac{1}{Ni}\sum_{i=1}^{Ni}(target(i) - output(i))^2} \quad (21)$$

Table 3. MSE vs No of nodes

Nodes	BPNN	GA	TwinSVM
100	0.1689	0.097864	0.061657
200	0.15133	0.10173	0.053477
300	0.13107	0.094877	0.047967
400	0.13092	0.092049	0.045128
500	0.12675	0.083861	0.034257
600	0.11185	0.079677	0.050193
700	0.11036	0.075515	0.025922
800	0.10887	0.056597	0.027104
900	0.091302	0.04974	0.017576
1000	0.081767	0.028143	0.0134
Average	0.121312	0.076005	0.0376681

Here, we consider the range from 100 to 1000 sensor nodes for data transmission. The table above shows the fault detection ability of the various approaches in terms of MSE. On average the BPNN approach has an MSE of 0.12, the GA approach has an MSE of 0.07 and the TwinSVM has an MSE of 0.037. This shows that the TwinSVM is 50.4% better than the GA based approach and 68.95% better than the BPNN based approach. In this simulation, it has been observed that by increasing the number of nodes, the TwinSVM performs better than the existing protocols. This can

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

be attributed to the fact that, the more samples we provide to the TwinSVM, the more effectively it is able to learn; thereby producing fewer errors, as depicted in Figure 5.

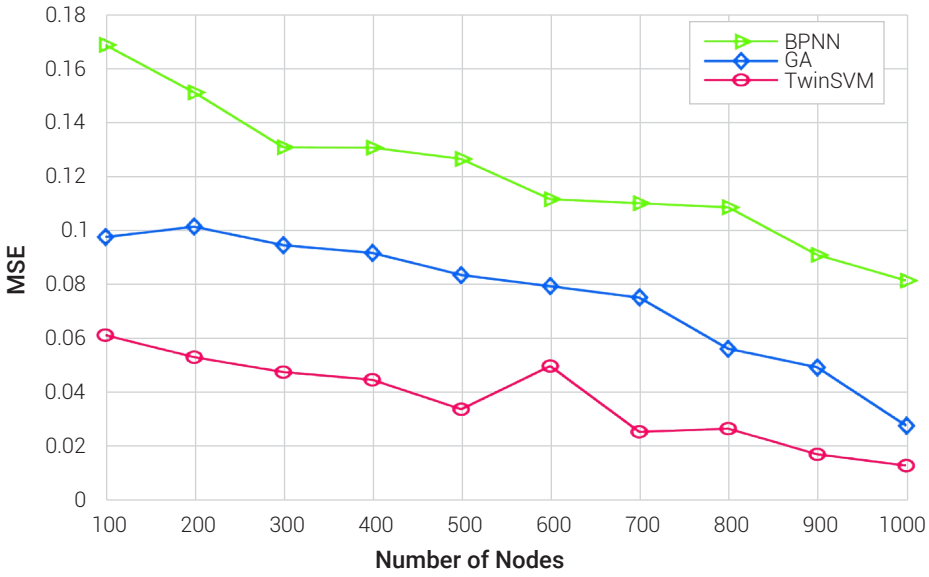

Figure 5. MSE vs Number of Nodes
Source: own work

3.2.2. Detection Accuracy versus Number of Nodes

The detection accuracy of the TwinSVM classifier can be calculated using Equation (22).

$$\boldsymbol{Detection\ Accuracy} = \frac{\boldsymbol{Identified\ Faulty}}{\boldsymbol{Total\ Faulty\ Nodes.}} \tag{22}$$

For example, there are a total of 275 nodes in the network that are are introduced as faulty from the previously defined range of 20-30% out of 1000 nodes for the simulation and the TwinSVM is able to identify 271 nodes. The detection accuracy will be calculated as below and in Figure 6:

$$\boldsymbol{Detection\ Accuracy} = \frac{\mathbf{271}}{\mathbf{275}} = \mathbf{98.54\%} \tag{23}$$

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

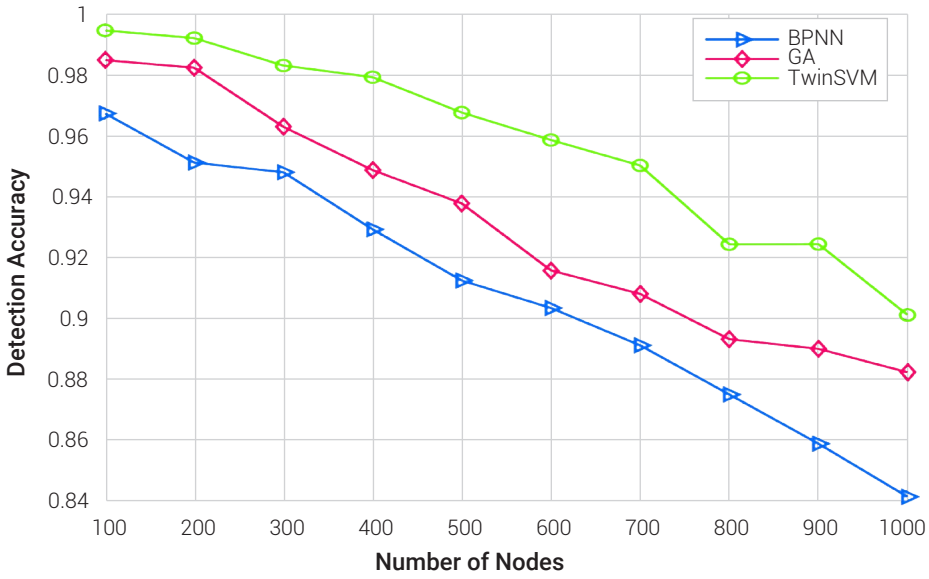

Figure 6. Detection accuracy versus Number of Nodes
Source: own work

The figure above shows detection accuracy of the various approaches. On average the BPNN approach has a DA of 90.8%, the GA approach has a DA of 93% and the TwinSVM has a DA of 95.7%. This shows that the TwinSVM is 2.7% better than the GA based approach and 4.9% better than the BPNN based approach. In this scenario, it has been observed that with an increase in the number of nodes, the TwinSVM performs better than the existing protocols. This can be attributed to the fact that the more samples we provide to the TwinSVM, the more effectively it is able to learn; thereby producing fewer errors.

3.2.3. False Classification Rate versus Number of Nodes

$$FCR = \frac{wrongly\ classified\ of\ the\ fault\ type}{total\ faulty\ in\ the\ fault\ type} \tag{24}$$

The average false classification rate for BPNN is 3%, for GD it is 4 %, and for TwinSVM it is 2%. This clearly signifies that the proposed protocol performs better than the existing protocols as in Figure 7.

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

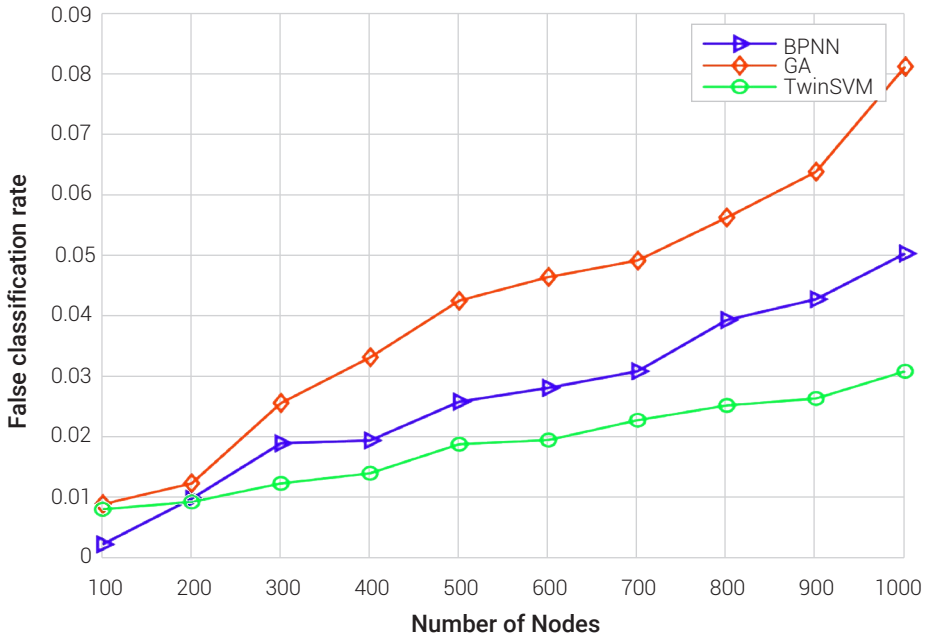

Figure 7. FCR versus Number of Nodes
Source: own work

False Alarm Rate versus Number of Nodes

The false alarm rate is the total number of normal nodes identified as faulty to the total number of normal nodes in the network and is calculated using Equation (25):

$$\boldsymbol{False\ Alarm\ rate} = \frac{\boldsymbol{Total\ number\ of\ false\ alarms}}{\boldsymbol{Total\ number\ of\ alarms}} \tag{25}$$

The average false alarm rate for BPNN is 4%, for GD it is 4%, and for TwinSVM it is 3%. This clearly signifies that the proposed protocol performs better than the existing protocols as in Figure 8.

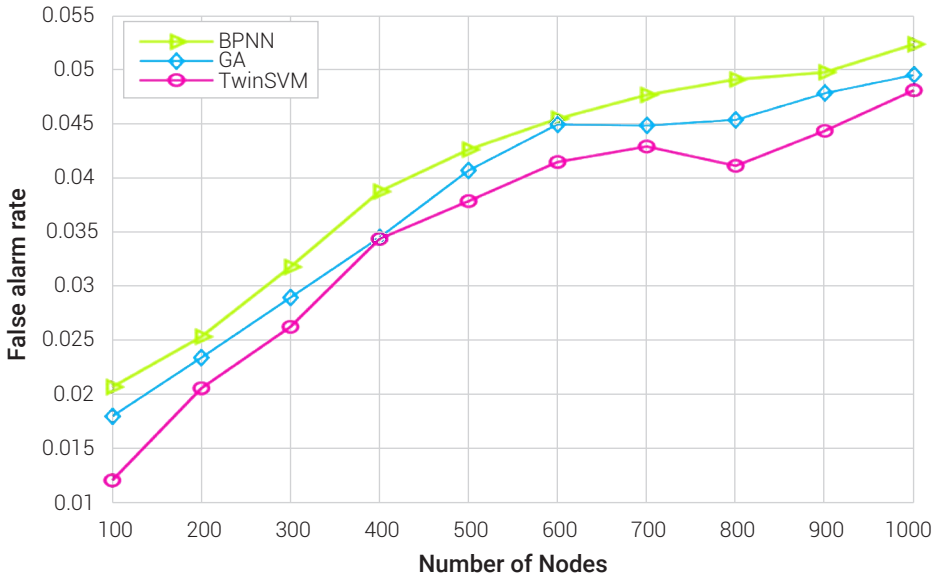

Figure 8. FAR versus Number of Nodes
Source: own work

3.2.5. False Positive Rate versus Number of Nodes

The false positive rate is the ratio between the number of faulty nodes identified as normal and the total number of faulty nodes in the network.

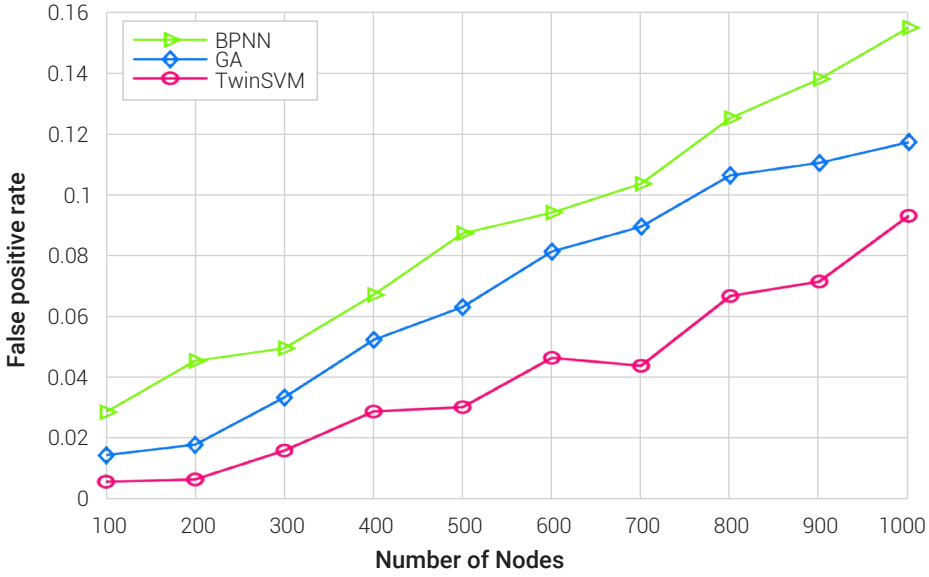

Figure 9. FPR versus Number of Nodes
Source: own work

The proposed fault diagnosis framework with TwinSVM performs better in terms of the generation of false positive rates as in Figure 9.

3.2.6. PDR versus Number of Nodes

As depicted in Figure 10, the average packet delivery ratio of the proposed fault diagnosis framework comes out to be 98.13%.

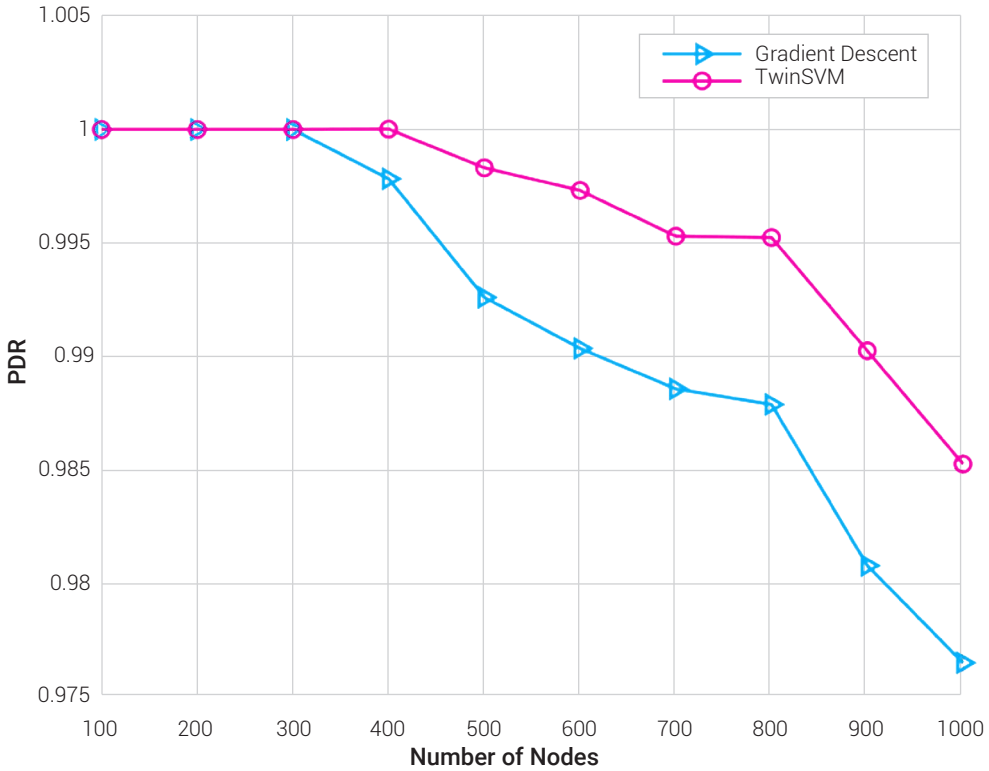

Figure 10. PDR versus Number of Nodes
Source: own work

3.2.7. Latency versus Number of Nodes

Latency constitutes the time complexity of the detection framework. Gradient decent and the GA approaches take longer to optimize, requiring more iterations than the TwinSVM, where support vectors in TwinSVM can be identified in single iterations, as depicted in Figure 11.

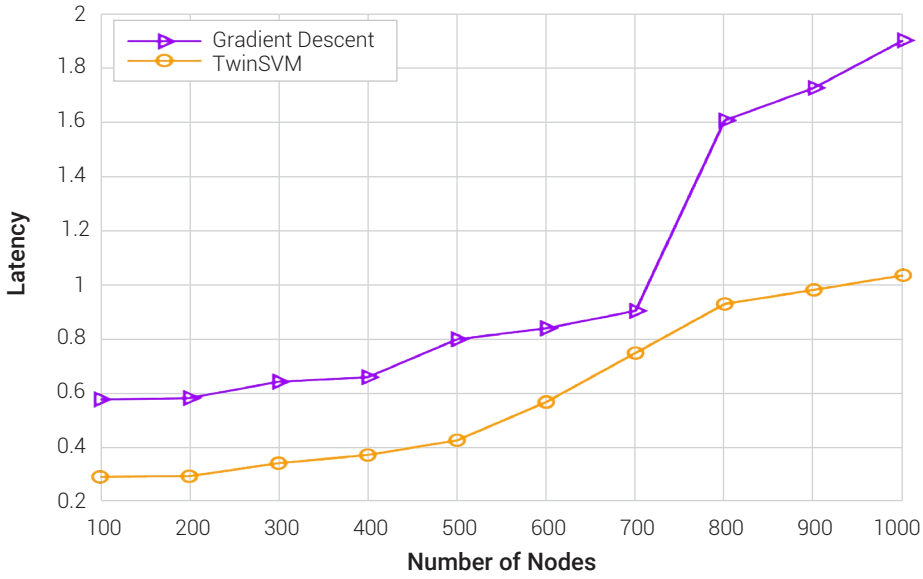

Figure 11. Latency versus Number of Nodes
Source: own work

3.2.8. Energy Consumption versus Number of Nodes

For the proposed framework and in the simulation where 20% of the faulty nodes are deployed ranging from 100 to 1000, Figure 12 below shows the level of energy consumption based on the number of nodes present.

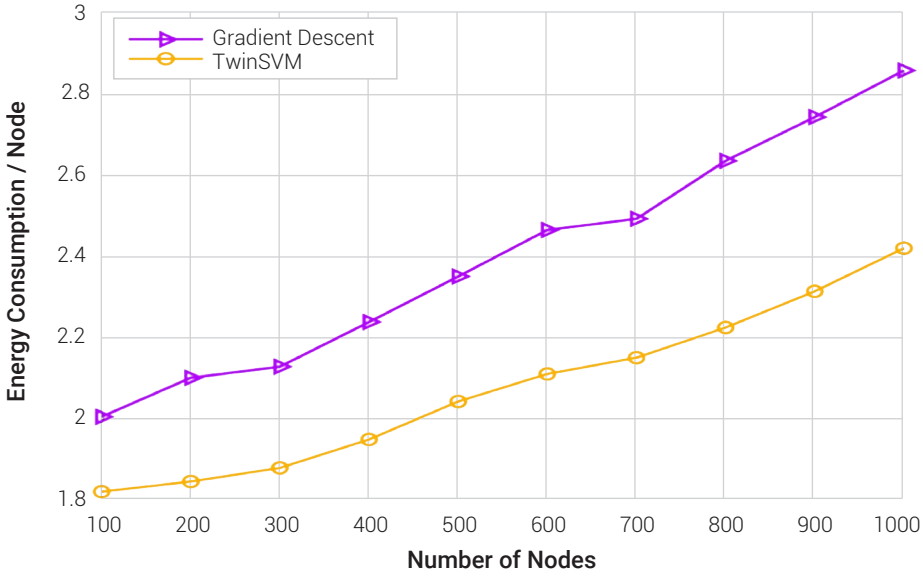

Figure 12. Energy consumption versus Number of Nodes
Source: own work

e-ISSN 2357-6014 / Vol. 18, no. 1 / january-april 2022 / Bogotá D.C., Colombia
Universidad Cooperativa de Colombia

On average, in GA, the average percentage of energy consumption is 2.4% but in the proposed scheme using TwinSVM the energy consumption is 2.07%, which clearly shows that the proposed framework consumes less energy when compared to the existing one; Since TwinSVM uses fewer equations to classify the faults compared to existing classifiers. Detection accuracies for the classifiers BPNN, GA and TwinSVM have been depicted in Table 4-5. As per the depicted values in the tables, it clearly signifies that the detection accuracy with 10% of faults being induced is greater when compared to other approaches; the trend remains the same in the effective accuracy, even in the presence of a greater number of faults, as in the case of 50% induction of faults.

Table 4. DA with 10% faults induced

DA with 10% Faults			
Fault Types	**TwinSVM**	**BPNN**	**GA**
Offset Fault	97.90%	93.90%	91.90%
Gain Fault	96.90%	90.90%	92.30%
Stuck-at Fault	95.90%	91.90%	95.70%
Out of Bounds	97.90%	98.90%	91.60%

Table 5. DA with 50% faults induced

DA with 50% Faults			
Fault Types	**BPNN**	**TwinSVM**	**GA**
Offset Fault	97.90%	94.30%	91.70%
Gain Fault	95.40%	94.30%	92.40%
Stuck-at Fault	94.60%	91.20%	94.50%
Out of Bounds	95.40%	96.50%	89.50%

4. CONCLUSION AND FUTURE SCOPE

Due to the mobile nature of the nodes in the MCS, all the mobile hosts and communication channels are prone to the occurrence of faults which may result in a deviated behavior of MCS from its desired normal behavior. In order to make the MCS fault tolerant, a fault detection and classification framework has been proposed to detect the type of faults in the MCS using TwinSVM. The proposed framework has been

tested against a number of parameters to evaluate the performance of the proposed system. The evaluation results generated better performance in terms of false positive rate, false alarm rate, detection latency, detection accuracy, energy consumption and packet drop ratio; as the detection accuracy has been considerably increased on an average of more than 90% in the proposed scheme, and packet drop ratio has been halved.

In future, the proposed framework can be tested using advanced support vector machine algorithms to evaluate the improvement in the efficiency of the MCS. As in this work, the main focus is the development of the fault detection framework along with fault classification. The obvious next phase of development will be the creation of a check pointing and rollback recovery method.

5. REFERENCES

[1] R. R. Swain and P. M. Khilar, "Composite Fault Diagnosis in Wireless Sensor Networks Using Neural Networks," *Wirel. Pers. Commun.*, vol. 95, no. 3, pp. 2507–2548, 2017. doi: https://doi.org/10.1007/s11277-016-3931-3

[2] E. Moridi, M. Haghparast, M. Hosseinzadeh, and S. J. Jassbi, "Fault management frameworks in wireless sensor networks: A survey," *Comput. Commun.*, vol. 155, pp. 205–226, 2020. doi: https://doi.org/10.1016/j.comcom.2020.03.011.

[3] S. Chessa and P. Santi, "Comparison-based system-level fault diagnosis in ad hoc networks," *Proc. 20th IEEE Symp. Reliab. Distrib. Syst.*, pp. 257–266, 2001. doi: https://doi.org/10.1109/RELDIS.2001.970776

[4] M. Elhadef, A. Boukerche, and H. Elkadiki, "A distributed fault identification protocol for wireless and mobile ad hoc networks," *J. Parallel Distrib. Comput.*, vol. 68, no. 3, pp. 321–335, 2008. doi: https://doi.org/10.1016/j.jpdc.2007.05.016

[5] T. Muhammed and R. A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 78, pp. 267–287, 2017. doi: https://doi.org/10.1016/j.jnca.2016.10.019

[6] S. Zidi, T. Moulahi, and B. Alaya, "Fault detection in wireless sensor networks through SVM classifier," *IEEE Sens. J.*, vol. 18, no. 1, pp. 340–347, 2018. doi: https://doi.org/10.1109/JSEN.2017.2771226

[7] M. Panda and P. M. Khilar, "Distributed Byzantine fault detection technique in wireless sensor networks based on hypothesis testing," *Comput. Electr. Eng.*, vol. 48, pp. 270–285, 2015. doi: https://doi.org/10.1016/j.compeleceng.2015.06.024

[8] U. Saeed, S. U. Jan, Y. D. Lee, and I. Koo, "Fault diagnosis based on extremely randomized trees in wireless sensor networks," *Reliab. Eng. Syst. Saf.*, vol. 205, no. October 2020, pp. 107284, 2021. doi: https://doi.org/10.1016/j.ress.2020.107284

[9] R. R. Swain, P. M. Khilar, and S. K. Bhoi, "Heterogeneous fault diagnosis for wireless sensor networks," *Ad Hoc Networks*, vol. 69, pp. 15–37, 2018. doi: https://doi.org/10.1016/j.adhoc.2017.10.012

[10] R. R. Swain, T. Dash, and P. M. Khilar, "A complete diagnosis of faulty sensor modules in a wireless sensor network," *Ad Hoc Networks*, vol. 93, pp. 101924, 2019. doi: https://doi.org/10.1016/j.adhoc.2019.101924

[11] S. K. Ghalem, B. Kechar, A. Bounceur, and R. Euler, "A probabilistic multivariate copula-based technique for faulty node diagnosis in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 127, pp. 9–25, 2019. doi: https://doi.org/10.1016/j.jnca.2018.11.009

[12] S. U. Jan, Y. D. Lee, and I. S. Koo, "A distributed sensor-fault detection and diagnosis framework using machine learning," *Inf. Sci. (Ny).*, vol. 547, pp. 777–796, 2021. doi: https://doi.org/10.1016/j.ins.2020.08.068

[13] C. Peng, Y. Zhou, and Q. Hui, "Distributed fault diagnosis of networked dynamical systems with time-varying topology," *J. Franklin Inst.*, vol. 356, no. 11, pp. 5754–5780, 2019. doi: https://doi.org/10.1016/j.jfranklin.2019.05.027

[14] S. Shao, S. Guo, and X. Qiu, "Distributed fault detection based on credibility and cooperation for wsns in smart grids," *Sensors (Switzerland)*, vol. 17, no. 5, pp. 983, 2017. doi: https://doi.org/10.3390/s17050983

[15] Z. Noshad *et al.*, "Fault detection in wireless sensor networks through the random forest classifier," *Sensors (Switzerland)*, vol. 19, no. 7, pp. 1–21, 2019. doi: https://doi.org /10.3390/s19071568

[16] N. Wang, J. Wang, and X. Chen, "A trust-based formal model for fault detection in wireless sensor networks," *Sensors (Switzerland)*, vol. 19, no. 8, pp. 1–20, 2019. doi: https://doi.org/10.3390/s19081916

[17] W. Hongying, D. M. Blough, and L. Alkalaj, "Analysis and experimental evaluation of comparison-based system-level diagnosis for multiprocessor systems," *Dig. Pap. - Int. Symp. Fault-Tolerant Comput.*, pp. 55–64, 1994. doi: https://doi.org /10.1109/FTCS.1994.315657

[18] I. G. A. Poornima and B. Paramasivan, "Anomaly detection in wireless sensor network using machine learning algorithm," *Comput. Commun.*, vol. 151, no. August 2019, pp. 331–337, 2020. doi: https://doi.org/10.3390/s21144946

[19] Jayadeva, R. Khemchandani, and S. Chandra, "Twin support vector machines for pattern classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 905–910, 2007. doi: https://doi.org/10.1007/978-981-13-1132-1_42

[20] H. Huang, X. Wei, and Y. Zhou, "Twin support vector machines: A survey," *Neurocomputing*, vol. 300, pp. 34–43, 2018. doi: https://doi.org/10.1016/j.neucom.2018.01.093

[21] K. J. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, no. 1–2, pp. 307–319, 2003. doi: https://doi.org/10.1016/S0925-2312(03)00372-2