Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2021.019082
Article

Tech Science Press

Fault-Tolerant Communication Induced Checkpointing and Recovery Protocol Using IoT

Neha Malhotra[1,2,*] **and Manju Bala**[3]

[1]I.K.Gujral Punjab Technical University, Kapurthala, 144603, India
[2]Lovely Professional University, Phagwara, 144411, India
[3]Khalsa College of Engineering and Technology, Amritsar, 143001, India
[*]Corresponding Author: Neha Malhotra. Email: mneha8789@gmail.com
Received: 01 April 2021; Accepted: 14 May 2021

Abstract: In mobile computing systems, nodes in the network take checkpoints to survive failures. Certain characteristics of mobile computing systems such as mobility, low bandwidth, disconnection, low power consumption, and limited memory make these systems more prone to failures. In this paper, a novel minimum process communication-induced checkpointing algorithm that makes full use of the computation ability and implementation of effective stable storage in a mobile computing system is proposed. The said approach initiates by taking spontaneous checkpoints by each node in phase 1 using a logistic function that is specifically used to estimate the time interval between two checkpoints and saves them locally. In phase 2, each node takes checkpoints in a coordinated manner using the Takagi–Sugeno (T–S) fuzzy system, which generates results based on the interpretation of 39 rules specifically incorporated in the system to avoid unnecessary and irrelevant checkpoints. Finally, the permanent checkpoints are stored on IoT(Internet of things) to reduce the storage capacity of the system. Quantitative analysis and experimental simulation prove that the proposed scheme outperforms other communication-induced checkpointing schemes in terms of the minimum number of processes required to take the checkpoints and communication cost. Simulation results prove that the checkpointing process becomes faster as compared to the existing techniques due to a decrease in the latency wrt to the number of nodes and storage of permanent checkpoints on IoT. With the advent of network partitioning in the proposed system, the average computation loss has also reduced as the rollback recovery takes place in a particular partition only, not in the entire network. The overall approach makes a mobile distributed computing system fault tolerant while non-blocking of the processes during the checkpointing process.

Keywords: Checkpointing; distributed system; mobile computing; IoT; fault tolerance

1 Introduction

The advancement in communication technology and its rapid development from wired to mobile networks enables access to services anytime and anywhere by such a system. The devices using such services are required to be equipped with resources so that the system can guarantee to operate even in the presence of faults. Communication links in such systems are required to be failure-prone for the smooth functioning of the system. Checkpointing-based recovery is a critical approach in making the system as fault tolerable [1,2]. Checkpointing is a technique that focuses on the confinement of faults in the system and its proper restoration from the stable states [3,4]. It is the process of saving a consistent state of the process in a system so that the system can be recovered from the last consistent checkpoint in case of fault. Most of the checkpointing techniques are not focusing on the stable storage concept, which is one of the most critical resource challenges in mobile computing. The existing checkpointing protocols cannot handle the challenges of mobile computing like, in mobile networks, resources are limited, and due to the mobile nature of the nodes, frequent disconnections and reconnections are possible.

Moreover, to store the checkpoints, consistent, stable storage and implementation are required, which has not been considered. To design an efficient checkpointing and recovery protocol, several points are required to be addressed like, to get a most consistent and recent global state of checkpoints, overall latency and computation cost of the messages should be less, must reduce the amount of energy and bandwidth consumed, capable of handling frequent disconnections and reconnections in the network and efficient mechanism to implement stable storage—the proposed protocol presented in this paper able to handle all these challenges. The study's objective is that the checkpoints were taken on a periodic time interval in each node in the existing system. We see a potential improvement if we can make it dynamic based on specific network environment conditions.

2 Related Work

Checkpointing approach can be classified into three different approaches: coordinated checkpointing approach, uncoordinated checkpointing approach, and communication induced checkpointing approach. In an uncoordinated approach, processes in the system can decide the time to take the checkpoint as they do not depend on other processes in the system. The main advantage of this approach is that the processes can reduce the overhead in the system by reducing the number of checkpoints to be taken. However, there are several disadvantages of this approach, and the most subtle one is the domino effect which can increase the overhead to a great extent in the system [5]. Secondly, processes may take multiple checkpoints, which may not be required for consistent state capturing. Thirdly, this approach requires garbage collection of all the new checkpoints. Using this approach requires more memory in the system, which is a limited resource in mobile computing. In Guermouche et al. [6], and uncoordinated checkpointing approach has been used to send deterministic apps. With the above usage, the domino effect can be avoided using a small set of application messages.

Moreover, the protocol claims that the entire process need not roll back whenever a failure occurs in the system. The said approach is helpful to address the burst access challenge. Another uncoordinated has been proposed in Biswas et al. [7], in which each mobile node is required to keep a counter of how many clusters it has traversed during a checkpointing interval. This counter is incremented whenever there is a change in the cluster, and if this counter exceeds a predefined threshold, a checkpoint is required to be saved. In this manner, a log has been developed to maintain this information required in failure. In a coordinated approach, checkpoints can be taken in coordination with other processes in the system, and due to this reason, the said approach is also called the asynchronous approach of taking checkpoints. Whenever a process takes a new checkpoint, the old one has to be deleted. Upon initiating the checkpointing algorithm, one process initiates it and is responsible for saving the consistent checkpoint on stable storage. In Mansouri et al. [8], authors have introduced an adaptive, coordinated, and non-blocking approach to achieve fault tolerance in a mobile computing environment. In this scheme, only a few clusters are required to checkpoints, not all of which result in no blocking processes. An adaptive fault-tolerant mechanism has also been introduced by Cao et al. [9], which has reduced the synchronization messages and the number of checkpoints.

Along with this, the approach has also made the checkpointing process non-blocking. Another technique presented in Benkaouha et al. [10] has been designed based on the cluster-based routing protocol, and the scheme ensures to take a minimum number of cluster-based checkpoints. The said scheme also reduces the control and computation messages while reducing energy consumption and latency. In the third approach, a combination of the above two approaches has been used. As in communication-induced checkpointing, processes take checkpoints in two phases locally and globally. Once in an uncoordinated manner at the node level and then in coordination with the partition manager. The main advantage of this approach is that there is no domino effect which eventually prevents cascading termination. Authors in Ahn [11] introduced an efficient protocol to detect Z-cycle free patterns but with no additional computational messages. This approach promises to take few checkpoints.

In Garcia et al. [12], reviews the class of communication induced checkpointing protocols like Fully Informed (FI) and Lazy-FI, FINE (Fully informed and efficient), and Lazy-FINE protocols, which are classified as the best approaches towards this class of checkpointing. In Abdelhafidi et al. [13], two communication induces protocols, namely CSFDAS and CSRDTP, that guarantee the RDT property while reducing the overhead messages. These protocols ensure to carry a contact size of control information which appended with the processes. Authors in Ahn [14] have introduced HMNR protocol to control information of each process can be controlled, which eventually results in minimal forced checkpoints. Also, the improved version of HMNR called Lazy-HMNR has been presented to lower the number of Z-cycles. A formal framework to express the operational semantics of the existing snapshot algorithms has been proposed in Kiehn et al. [15], which classifies the coordinated algorithm with communication-induced algorithms. Since in literature, many checkpointing algorithms presented were expressed using pseudo code only, hence authors have provided a framework for the evaluation.

3 System Model and Assumptions

In the proposed mobile computing system, nodes in the system have been divided into several partitions, and each partition has its partition manager to look up for all the members of that partition. Partition managers communicate with each other through particular nodes called gateways. Following assumptions have been made to carry out the process:

First, nodes in the system take checkpoints in two phases, once locally and then globally. Second, channels in the system are reliable that results in no message lost during the communication. Third, better connectivity has been possessed by the nodes in the system, making these nodes enable to exchange messages before disconnection and post reconnection to the network. Moreover, the T-S fuzzy system helps to take forced checkpoints effectively.

4 Proposed Work

There are several protocols discussed in the literature. However, the main work of our study focuses on a hybrid approach of checkpointing, which generally takes place in two phases: Phase1: in this phase, each process in the system takes spontaneous checkpoint without coordinating with other processes and save it in its local storage which eventually prevents storing a large number of checkpoints on stable storage that further reduces the amount of energy and bandwidth consumed, Phase2: in this phase, checkpoints have been taken in coordination with the other processes in the system. The partition manager is the initiator, initiates the process to record the consistent global checkpoint on stable storage while overcoming the domino effect. To achieve this, the partition manager broadcasts to its member nodes to take checkpoints which further propagate the same request to their neighbors. The most consistent and recent local checkpoint, which has already been saved with each node during the phase1, has been transferred to the stable storage so that a globally consistent and permanent checkpoint has been determined. Two parameters have been initialized before initiating the checkpointing process, 'α' before phase 2 starts, the count of messages taken by the nodes locally are represented using this, and 'T' is the time to decide when to take the next checkpoint. Finally, the tentative checkpoints saved in IoT as permanent ones depicted in Fig. 1. CP defines the checkpoints, MH defines the mobile host, next(i) is the counter that increments after every 'T' time units, Sn(i) is the sequence number of the last checkpoint.

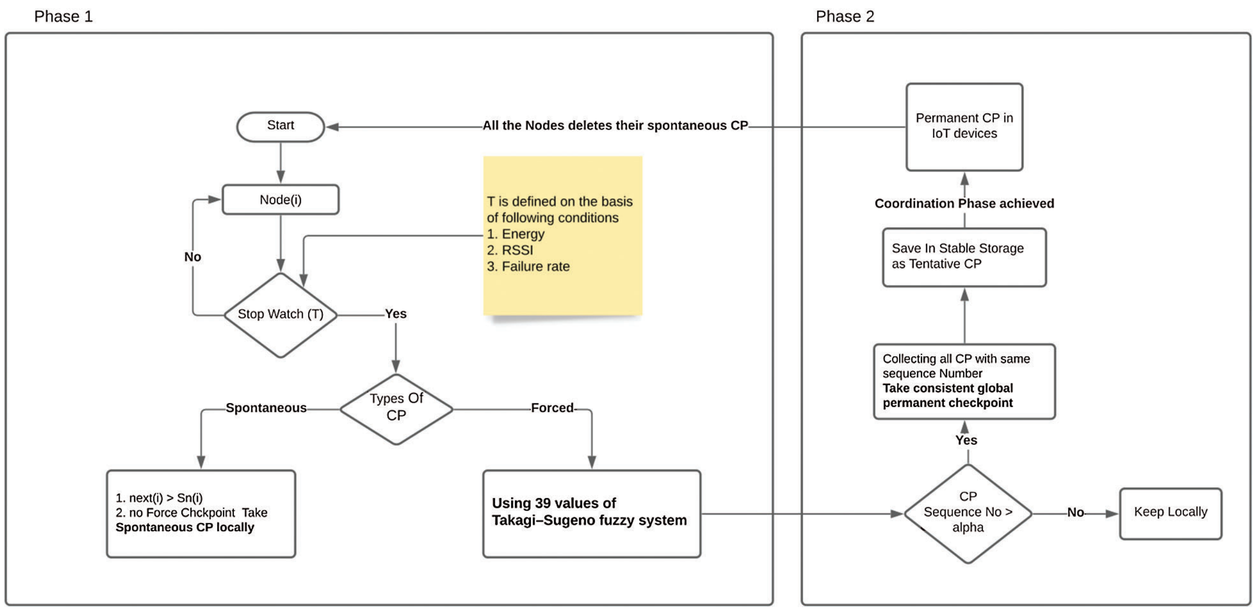
Phase 1
Phase 2
Start
All the Nodes deletes their spontaneous CP
Permanent CP in IoT devices
Node(i)
T is defined on the basis of following conditions
1. Energy
2. RSSI
3. Failure rate
No
Stop Watch (T)
Yes
Coordination Phase achieved
Save In Stable Storage as Tentative CP
Types Of CP
Spontaneous
Forced
Collecting all CP with same sequence Number
Take consistent global permanent checkpoint
Yes
1. next(i) > Sn(i)
2. no Force Chckpoint Take Spontaneous CP locally
Using 39 values of Takagi–Sugeno fuzzy system
CP Sequence No > alpha
No
Keep Locally

Figure 1: Step by step flow diagram for the proposed model

4.1 Phase1 (Take Spontaneous Checkpoints Using Logistic Function)

As each node takes checkpoints and saves them locally, then these inculcate the need to store the sequence number of the checkpoints and the variable to decide the time interval for each checkpoint. In our work, we have introduced the concept of logistic function to estimate the time intervals between two checkpoints. To accomplish the same following parameters have been taken: Sni = Sequence number of each checkpoint taken by the nodes and Ctri = Counter, which is incremented every time to decide whether to take a random checkpoint or not. Information of the parameters mentioned above has been maintained by each of the nodes in the network so that the same information can be piggybacked in the computation messages to be shared with other nodes in the network. Based on the value of sequence number and logistic functions, nodes will decide to take the next checkpoint in the sequence. The logistic function is used for estimating the time interval between the random checkpoints, The equation for the logistic function-based checkpointing is given below. This function works by first identifying the parameters T_max, β, α and α0 as in (1) and in Fig. 2:

$$\mathrm{T(checkpoints)} = \frac{\mathrm{T_{max}}}{1 + \mathrm{e}^{-\beta(\alpha-\alpha 0)}} \tag{1}$$

T_{max} = the maximum time interval between two checkpoints, 'e' is exponential, '$\alpha 0$' is the logistic function midpoint meaning at what checkpoint number the curve will become exponential, 'β' = growth rate or the steepness of logistic function taking random checkpoints, 'α' maximum checkpoints that can be taken.

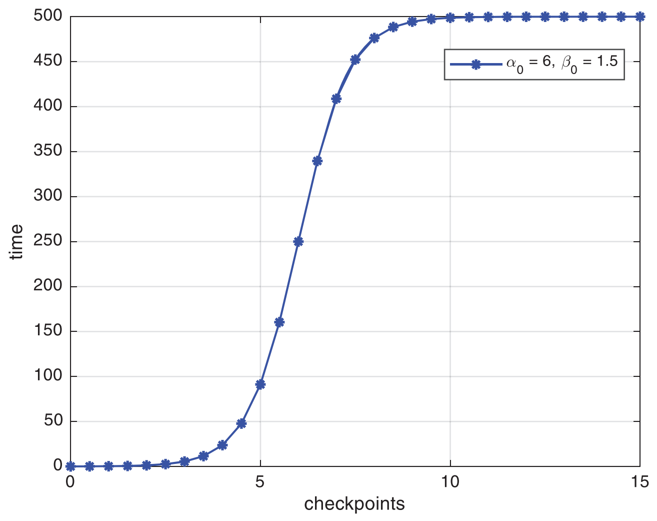

Figure 2: Time Interval for taking spontaneous checkpoint

The logistic function starts near 0, i.e., the node must take the first checkpoint immediately at first grows slowly initially till the current number of checkpoints near $\alpha 0$, then grows very fast to reach T_{max} and eventually levels off and converges to the T_{max}, i.e., the maximum allowed time interval between two checkpoints as depicted in Fig. 3.

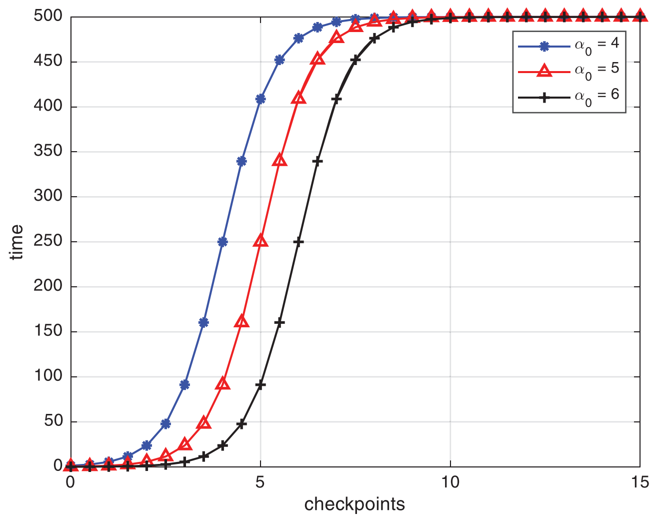

Figure 3: Behavior of the curve at varying $\alpha 0$ and fixed β

α0 and β control the shape of the curve as shown in figure changing α0 can make the initial checkpoint interval slower and faster; however, in all cases, the curve converges to T_{max} regardless of the no of checkpoints or the growth rate β as shown in Fig. 4 below.

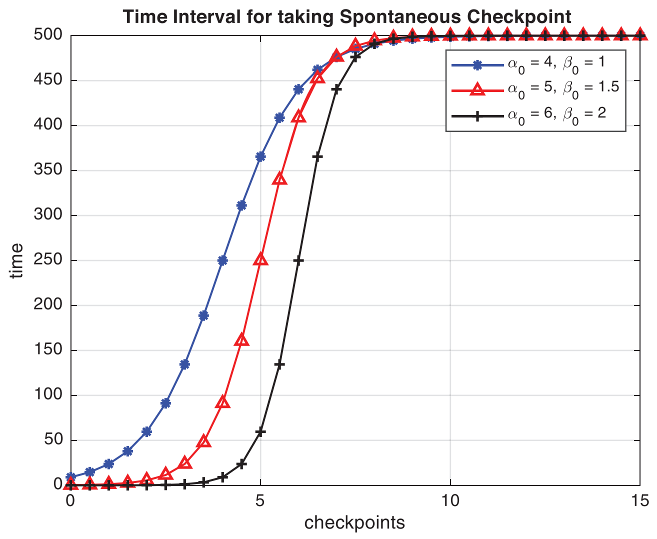

Figure 4: Behavior of the curve at varying α0 and β

4.2 Phase 2 (Coordination Phase)

As the forced checkpointing decision depends on various scenarios, the overall goal is to reduce the number of forced checkpoints and improve coordination. Taking checkpoints only when one node in the system has performed its task leaves out many situations when forced checkpointing is needed; for instance, what happens if the energy of one node drops below usable limits or the node moves out of the range of the nearby coordinator. In these situations, the whole partition will be forced to go back to the last checkpoint instead of the node suffering from these conditions; this affects the system as all the nodes need to take a forced checkpoint when one node is either have completed work or forces all nodes back to the last checkpoint when one node goes down [16]. This has the consequences like, it introduces overall network overhead, it incurs extra latency forced checkpoints are taken more often, it decreases the overall reliability of the network, and it makes the overall system pseudo-distributed.

A Takagi–Sugeno (T–S) fuzzy system has been taken into consideration for phase 2 of checkpointing as depicted in Fig. 5, having three inputs *a1*, a2, and a3 an output ***b*** described by ***r*** fuzzy inference rules, some of the rules taken for consideration are depicted in the table below. The reasoning for considering T-S-based fuzzy systems is because they are more general than the Mamdani fuzzy systems. The T-S systems are better as they allow parallel distributed control. A parallel distributed control is fundamental in the development of any distributed checkpointing system.

T–S system is described by “IF-THEN” rules which represent local input/output relations. The main characteristic of a T–S fuzzy model concerns the partitioning of a complex system with the help of input models for capturing the system’s overall complexity obtained by the fuzzy blending of the set of inputs. As the coordination phase can become complex very fast and ensure the overall reliability of the system, a coordination phase for forced checkpointing using the T-S system based on Energy, Failure, and RSSI is used, as depicted in Fig. 4 above. These specially selected parameters help nodes in the distributed system decide when to take a forced checkpoint depicted in Tab. 1 below:

$$K_j\text{: IF } a_1 \in M_j(a_1) \text{ AND } a_2 \in N_j(a_2) \text{AND } a_3 \in O_j(a_3), \text{ then } b = P_j(a_1, a_2, a_3) \tag{2}$$

where j = 1,2,…,r, Mj(a1), Nj(a2), Oj(a3), are fuzzy sets, and Pj (a1, a2, a3) is the polynomial of degree d. degree d defines degree the of membership to that particular fuzzy variable. The inputs of the system are the Remaining energy of the node, Node's Failure Rate, and RSSI from the partition manager, the ranges for defining the membership function are depicted in Tab. 2 below:

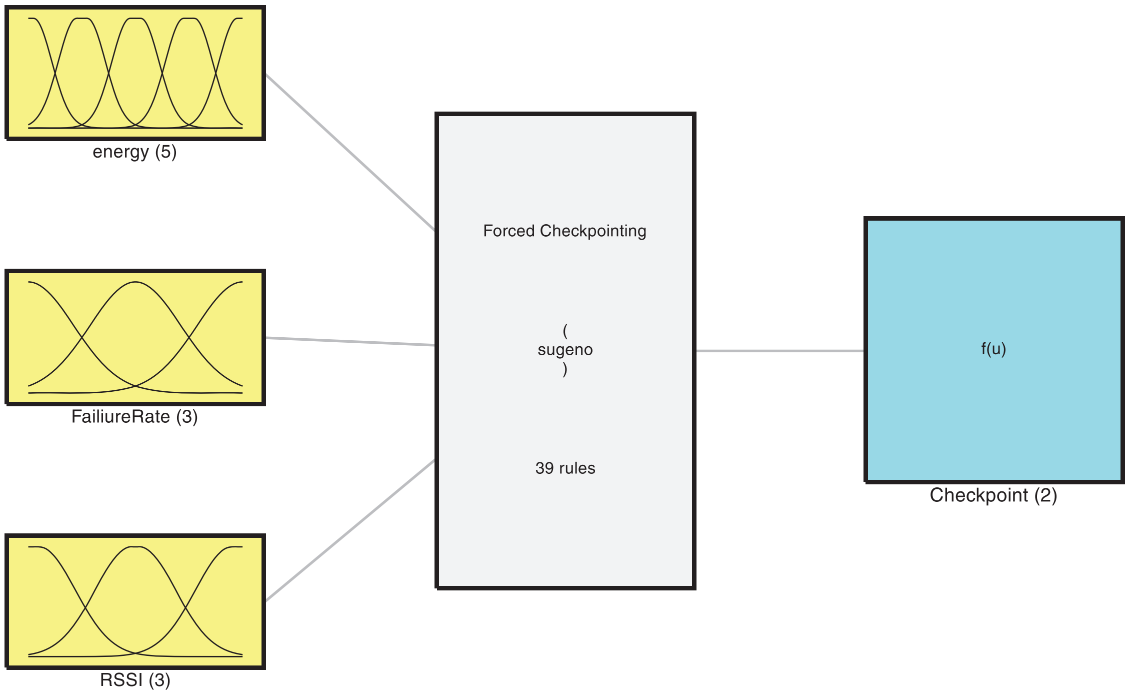

Figure 5: Takagi–Sugeno (T–S) fuzzy system for forced checkpointing

Table 1: Few rules of checkpointing in the Takagi-Sugeno fuzzy system

Energy	Failure Rate	RSSI	Checkpoint
Very Low	Low	Unusable	immediate
Low	Low	Normal	Normal
Mid	High	Unusable	immediate
High	Mid	High	Normal
Very High	Mid	Unusable	immediate

Table 2: Ranges of membership functions

Parameter	Range
Remaining Energy	0-1 joules
Failure Rate	0-1
RSSI	–120-0 db

For the implementation of fuzzy membership rules, we have used Gaussian membership functions and are defined by (3), (4), and (5); Gaussian MFs are because of their smoothness and concise notation. These curves have the advantage of being smooth and nonzero at all points. Also, Gaussian functions are local but not strictly compact, the desired output is very smooth, and Multivariate Gaussian functions can be formed using different inputs as we have used in our case Remaining Energy of node, Failure Rate of node, and RSSI, as depicted in Figs. 6, 7 and 8 respectively.

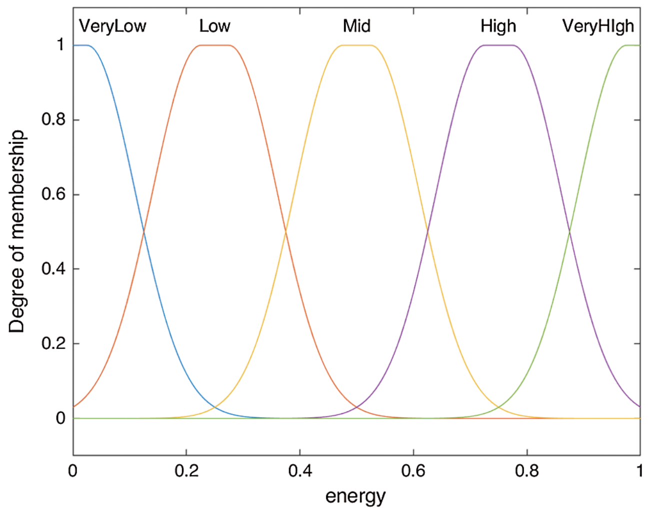

Figure 6: Membership function based on the remaining energy of the node

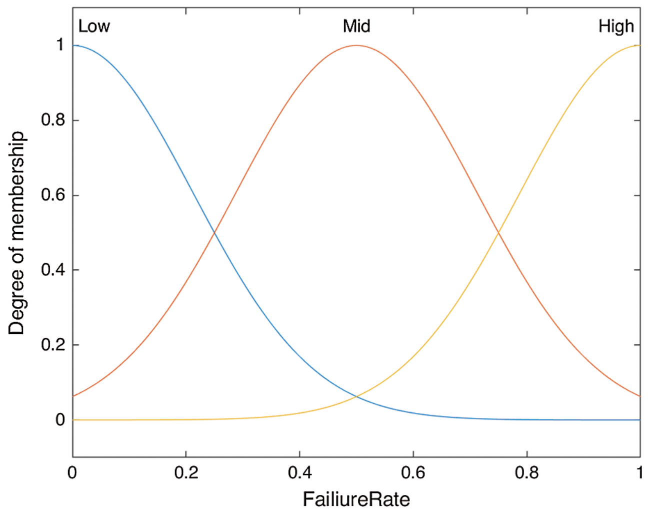

Figure 7: Membership function based on the failure rate of the node

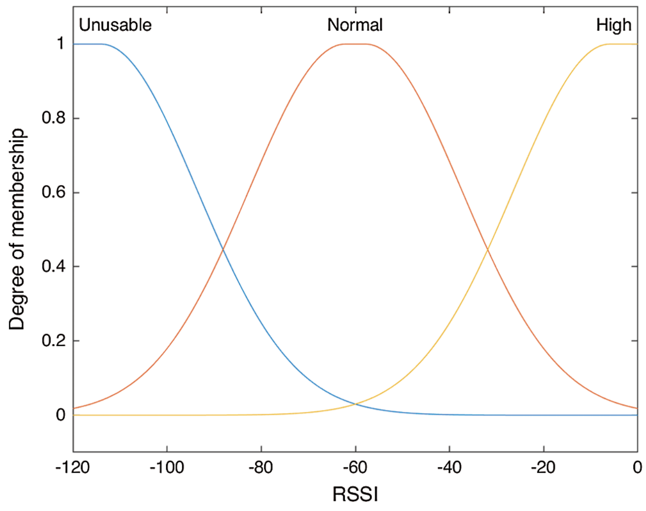

Figure 8: Membership function based on the RSSI from the partition manager of the node

$$\boldsymbol{S}_1(\boldsymbol{a}_1) = \mathbf{gauss}(\boldsymbol{a}_1; \boldsymbol{p}_1, \boldsymbol{\sigma}_1) = \exp\left(-\frac{1}{2}\left(\frac{\boldsymbol{a}_1 - \boldsymbol{p}_1}{\boldsymbol{\sigma}_1}\right)^2\right) \tag{3}$$

$$\boldsymbol{P}_1(\boldsymbol{a}_2) = \mathbf{gauss}(\boldsymbol{a}_2; \boldsymbol{q}_1, \boldsymbol{\delta}_1) = \exp\left(-\frac{1}{2}\left(\frac{\boldsymbol{a}_2 - \boldsymbol{q}_1}{\boldsymbol{\delta}_1}\right)^2\right) \tag{4}$$

$$\boldsymbol{R}_1(\boldsymbol{a}_3) = \mathbf{gauss}(\boldsymbol{a}_3; \boldsymbol{s}_1, \partial_1) = \exp\left(-\frac{1}{2}\left(\frac{\boldsymbol{a}_3 - \boldsymbol{s}_1}{\partial_1}\right)^2\right) \tag{5}$$

The output of the T–S system is computed by (6):

$$\boldsymbol{b} = \frac{\sum_{j=1}^{r} \boldsymbol{M}_j(\boldsymbol{a}_1)\boldsymbol{N}_j(\boldsymbol{a}_2)\boldsymbol{O}_j(\boldsymbol{a}_3)\boldsymbol{P}_j(\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3)}{\sum_{j=1}^{r} \boldsymbol{M}_j(\boldsymbol{a}_1)\boldsymbol{N}_j(\boldsymbol{a}_2)\boldsymbol{O}_j(\boldsymbol{a}_3)} \tag{6}$$

The *fuzzy basis function* (FBF) for the *j*th rule is given by (7):

$$\xi_j(\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3) = \frac{\boldsymbol{M}_j(\boldsymbol{a}_1)\boldsymbol{N}_j(\boldsymbol{a}_2)\boldsymbol{O}_j(\boldsymbol{a}_3)}{\sum_{j=1}^{r} \boldsymbol{M}_j(\boldsymbol{a}_1)\boldsymbol{N}_j(\boldsymbol{a}_2)\boldsymbol{O}_j(\boldsymbol{a}_3)} \tag{7}$$

Applying (7), the output of the T–S system can be written as (8) for the zero-order system:

$$\boldsymbol{b} = \sum_{j=1}^{r} \xi_j(\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3)\boldsymbol{x}_j \tag{8}$$

And for the first-order and high-order systems as (9):

$$\begin{aligned} b =& \sum_{j=1}^{r} \xi_j(a_1,a_2,a_3)a_1^m w_{mj} + \ldots + \xi_j(a_1,a_2,a_3)a_1 w_{1j} \\ &+\xi_j(a_1,a_2,a_3)a_2^m v_{mj} + \ldots + \xi_j(a_1,a_2,a_3)a_2 v_{1j} \\ &+\xi_j(a_1,a_2,a_3)a_3^m u_{mj} + \ldots + \xi_j(a_1,a_2,a_3)a_3 u_{1j} \\ &+\xi_j(a_1,a_2,a_3)x_j \end{aligned} \tag{9}$$

As in (9) the fuzzy basic functions are multiplied by $a1l$, $a2l$, $a3l$ where $l = 1, 2, \ldots, m$. The *modified FBF* (MFBF) for the jth rule is the function $hl\,j$ ($a1$, $a2$, $a3$) or $gl\,j$ ($a1$, $a2$, $a3$) or $kl\,j$ ($a1$, $a2$, $a3$) given by (10), (11) and (12):

$$h_{lj}(a_1,a_2,a_3) = \xi_j(a_1,a_2,a_3)a_1^l \tag{10}$$

$$g_{lj}(a_1,a_2,a_3) = \xi_j(a_1,a_2,a_3)a_2^l \tag{11}$$

$$k_{lj}(a_1,a_2,a_3) = \xi_j(a_1,a_2,a_3)a_2^l \tag{12}$$

After applying (10), (11) and (12), (13) has been obtained:

$$\begin{aligned} b =& \sum_{j=1}^{r} h_{mj}(a_1,a_2,a_3)w_{mj} + \ldots + h_{1j}(a_1,a_2,a_3)w_{1j} \\ &+g_{mj}(a_1,a_2,a_3)v_{mj} + \ldots + g_{1j}(a_1,a_2,a_3)v_{1j} \\ &+k_{mj}(a_1,a_2,a_3)u_{mj} + \ldots + k_{1j}(a_1,a_2,a_3)u_{1j} \\ &+\xi_j(a_1,a_2,a_3)x_j \end{aligned} \tag{13}$$

For zero order, defined in (14);

$$\begin{aligned} h_j(a_1,a_2,a_3) &= \xi_j(a_1,a_2,a_3), \\ w_j &= x_j \end{aligned} \tag{14}$$

For first order, defined in (15);

$$\begin{aligned} h_j(a_1,a_2,a_3) &= \left[h_{mj},\ldots,h_{1j},g_{mj},\ldots,g_{1j},k_{mj},\ldots,k_{1j}\xi_j\right] \\ w_j &= \left[w_{mj},\ldots,w_{1j},v_{mj},\ldots,v_{1j},u_{mj},\ldots,u_{1j}x_j\right]^T \\ &\text{where } \dim(h_j) = \dim\left(w_j^T\right) = 2d+1 \end{aligned} \tag{15}$$

The output can we rewritten as (16) and (17):

$$\begin{aligned} b &= [h_1(a_1,a_2,a_3),\ldots,h_r(a_1,a_2,a_3)]\begin{bmatrix} w_1 \\ \vdots \\ w_r \end{bmatrix} \\ &= h(a_1,a_2,a_3)w \end{aligned} \tag{16}$$

$$\begin{aligned} h(a_1,a_2,a_3) &= [h_1(a_1,a_2,a_3),\ldots,h_r(a_1,a_2,a_3)], \\ w &= [w_1,\ldots,w_r]^T \end{aligned} \tag{17}$$

Fig. 9 above shows the output spread of output $\boldsymbol{b}$ described by $\boldsymbol{r}$ fuzzy inference rules, areas in red show the condition for the forced checkpoint produced by the T-S fuzzy system. As the energy of the node drop below 20%, the node is forced to take a checkpoint at the stable storage. Similarly, if the node has a higher failure rate, the node is forced to take checkpoints at the stable storage more frequently.

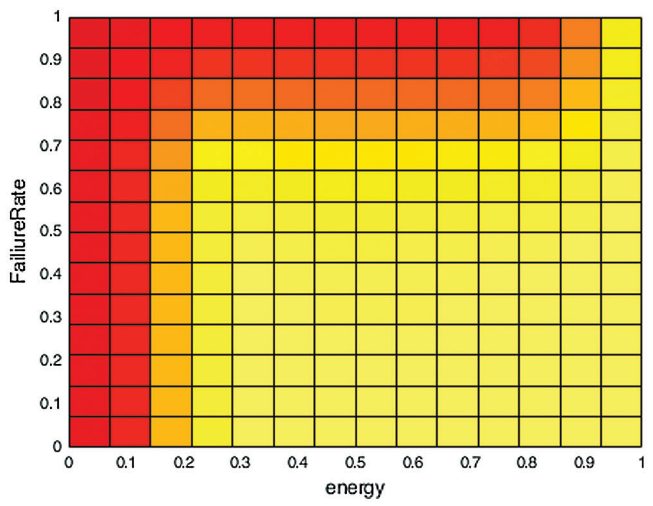

Figure 9: Effect of remaining energy and failure rate of the node on the decision to take a checkpoint

4.3 Storage of Permanent Checkpoints in IoT

To end the second phase of taking checkpoints, an already defined method by authors in Cao et al. [9] has been used in which a variable is set to 1 in phase 1 by the initiator. Before sending the request, any node divides that this variable by 2 for each of the recipients. To its response, the remaining value of this variable is shared [17–20]. The initiator of phase 1 sums these values upon receiving the responses, and if the total of the sum comes out to be one, then the second phase ends while converting the tentative checkpoints to permanent checkpoints on IoT and asks other nodes in the network to do the same. This process is repeated, and all the nodes delete their spontaneous and forced checkpoints whose sequence number is less than the permanent checkpoint saved on IoT.

5 Simulation Parameters

Simulation of the proposed work is carried out in MATLAB by checking the effect of the increasing value of ‘α’, i.e., the number of checkpoints and number of checkpoints taken in IoT. Following parameters have been considered in Tab. 3 to carry out the same:

6 Performance Evaluation

Performance of the proposed work has been carried out using the parameters, checkpoints taken per node; control messages count, latency of the system, and overhead of messages per node. Initially, the performance is evaluated by determining how the method works if there is a change in the values of ‘α’. Results generated have been given in Fig. 10, and Tab. 4, which depicts that number of nodes has no impact on how many checkpoints have been taken, but the higher the value of ‘α’ lesser the coordination among nodes. However, if the value of ‘α’ is less [21], several control messages are more, which results in congestion. Hence, the figure shows that the proposed method can provide better results in any type of application only by setting ‘α’ and stores the permanent checkpoints onto stable storage.

Table 3: Simulation parameters

Area of network	1000 * 1000 m
Communication range	250 m
Nodes number	10-100
Mobility categorization	Low, high, and medium
Time of simulation	1000 s
Mobility model	Random waypoint mobility model
Fuzzy system	T-S system
Tmax	500 ms
Beta	1.5
Threshold	Alpha
Bandwidth (for IoT connection)	100 Mbps
Storage available in IoT	1 GB

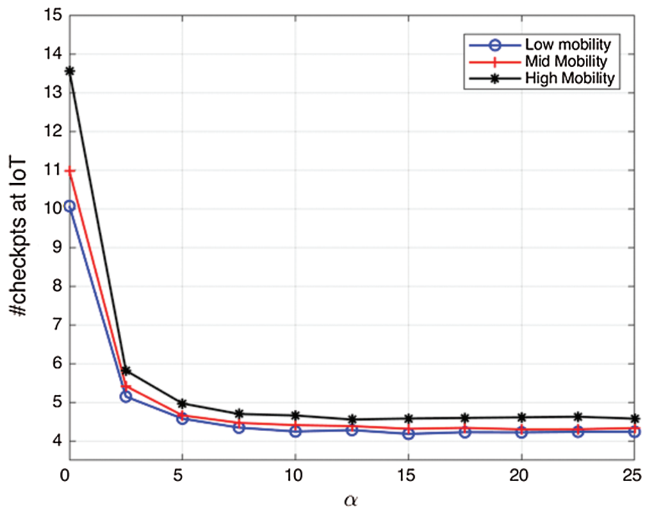

Figure 10: Number of checkpoints stored in IoT *vs*. taken

Table 4: Number of checkpoints taken in IoT

25 Nodes	50 Nodes	100 Nodes
10.091	10.968	13.507
5.2297	5.4059	5.859
4.5344	4.6282	4.91
4.3329	4.4268	4.6865
4.2261	4.3415	4.5739
4.2259	4.2718	4.5864

The following parameters used to access the performance of the proposed method are latency and control messages overhead. As shown in Figs. 11 and 12, the latency started decreasing after some value, mainly due to the more number of nodes as if the number of nodes is large, the possibility of message transfer is more that eventually results in a faster-checkpointing process.

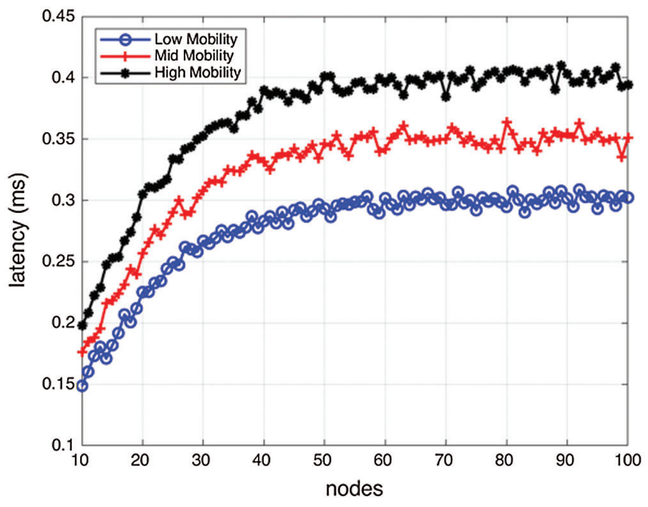

Figure 11: Impact of mobility on latency

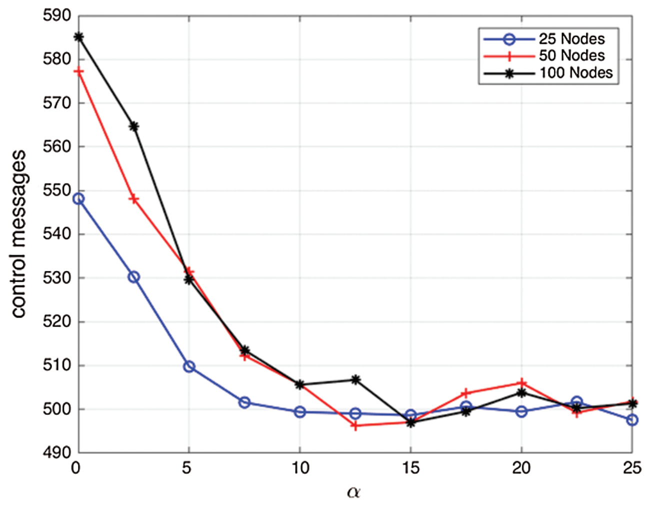

Figure 12: Impact of α on coordination control messages

Overhead of messages per node increases with the nodes, which is on an average 23 in low mobility, 25.5 in mid mobility, and 28 in high mobility. However, the mode of message propagation helps to simulate an exponential increase which is depicted in Fig. 13:

The performance of the proposed method has been measured on recovery by calculating the computation loss, which is depicted in Fig. 14. It is evident from the figure that due to partitioning in the network, rollback recovery occurs only in that partition, affecting the complete network and its functionality.

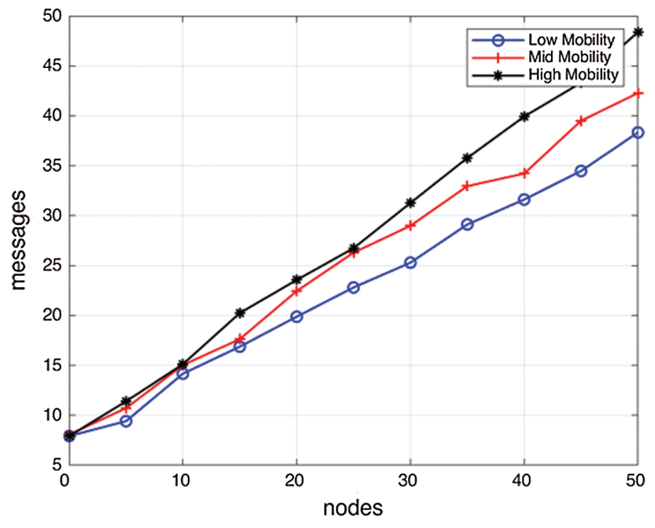

Figure 13: Number of the overhead of messages w.r.t. number of nodes at low, mid, and high mobility

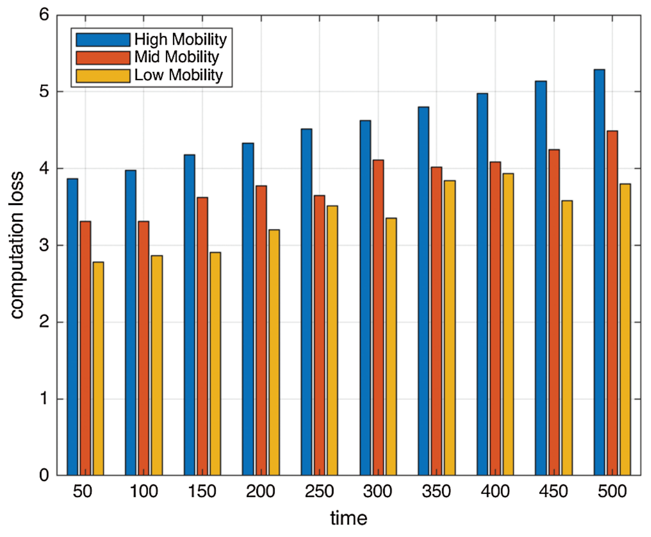

Figure 14: Average computation loss concerning time

As depicted in Fig. 15, it states how the records in IoT get impacted with disconnections which eventually increases the records in IoT.

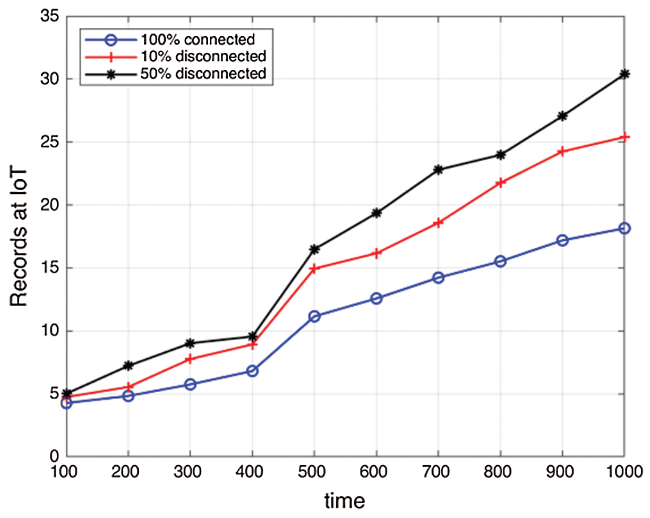

Figure 15: Impact of disconnections on records in IoT

7 Conclusion

A minimum process communication induced checkpointing algorithm with IoT has been presented to achieve fault tolerance in a mobile distributed computing system which ensures the non-blocking of processes during the checkpointing process. The proposed system helps to reduce the number of checkpoints locally taken by mobile nodes and implements stable storage in IoT. The said approach reduces the system overhead by minimizing the congestion of control messages among nodes. Moreover, the dependency of nodes has been reduced as if one node is at fault; then it will not affect the other nodes in the system to be rolled back. As a future direction, the proposed method can be implemented on other distributed applications. Evaluation of the system can be done on various security threats like denial of service attacks etc. Also, the impact of the proposed can be diagnosed on heterogeneous environments and onto real-time hardware scenarios.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no interest in reporting regarding the present study.

References

[1] S. Biswas and S. Neogy, "Secured fault-tolerant mobile computing," *in Proc. Communication in Computer and Information Science*, vol. 191 CCIS, no. PART 2, pp. 417–429, 2011.

[2] C. Men, Z. Xu and D. Wang, "An efficient handoff strategy for Mobile Computing checkpoint system," in *Proc. Embedded and Ubiquitous Computing, LNCS*, pp. 410–421, 2007.

[3] S. Biswas and S. Neogy, "A mobility-based checkPointing protocol for mobile computing system," *International Journal of Computer Science and Information Technology (IJCSIT)*, vol. 2, no. 1, February 2010.

[4] S. Biswas, P. Dey and S. Neogy, "Secure checkpointing-recovery using trusted nodes in MANET," in *Proc. - 4th IEEE Int. Conf. on Computer Communication Technology ICCCT 2013*, pp. 174–180, 2013.

[5] R. Tuli and P. Kumar, "New paradigms in checkpoint processing and recovery techniques for distributed mobile systems," *Proc. Trends in Network and Communications, Communications in Computer and Information Science CCIS*, vol. 197, pp. 221–231, 2011.

[6] A. Guermouche, T. Ropars, E. Brunet, M. Snir and F. Cappello, "Uncoordinated checkpointing without domino effect for send-deterministic MPI applications," in *Proc. - 25th IEEE Int. Parallel and Distributed Processing Symposium, IPDPS*, pp. 989–1000, 2011.

[7] S. Biswas and S. Neogy, “Mobility based checkpointing and trust-based recovery in manet,” *International Journal of Wireless & Mobile Networks*, vol. 4, pp. 53–69, 2012.

[8] H. Mansouri, N. Badache, M. Aliouat and A. S. K. Pathan, “Checkpointing distributed application running on mobile ad hoc networks,” *International Journal of High-Performance Computing and Networking*, vol. 11, no. 2, pp. 95–107, 2018.

[9] G. Cao and M. Singhal, “On coordinated checkpointing in distributed systems,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 12, pp. 1213–1225, Dec. 1998.

[10] H. Benkaouha, N. Badache, A. Abdelli, L. Mokdad and J. Ben Othman, “A novel hybrid protocol of checkpointing and rollback recovery for flat MANETs,” *International Journal of Automation and Adaptive Communication System*, vol. 10, no. 1, pp. 114, 2017.

[11] J. Ahn, “Efficient communication induced checkpointing protocol for broadcast network-based distributed systems,” *Journal of Distributed and Parallel Processing*, vol. 29, no. 1, pp. 1–12, 2019.

[12] I. C. Garcia, G. M. D. Vieira and L. E. Buzato, “A rollback in the history of communication induced checkpointing,” submitted(arXiv:1702.06167 [cs. DC]), Feb. 2017.

[13] Z. Abdelhafidi, N. Lagraa, M. B. Yagoubi and M. Djoudi, “Low overhead communication-induced checkpointing protocols ensuring rollback-dependency trackability property,” *Concurrency Computation Practice and Experience*, vol. 29, pp. e4271, 2017.

[14] J. Ahn, “Reducing the overhead of distributed checkpointing with group communication,” *Journal of advanced information technology and convergence*, vol. 10, no. 2, pp. 83–90, 2020.

[15] A. Kiehn and D. Aggarwal, “A study of mutable checkpointing and related algorithms,” *Science of Computer Programming*, vol. 1, pp. 1–15, 2017.

[16] N. Malhotra and M. Bala, “Fault Diagnosis in Wireless Sensor Networks-A Survey,” in *Proc.4th Int. Conf. on Computing Sciences (ICCS)*, IEEE, pp. 28–34, 2018.

[17] X. Huang, R. F. Rojas, A. C. Madoc and D. Ahmad, “Evidentiary assessment for protecting WSNs from internal attacks in real-time,” *Int. J. Comput. Appl*, vol. 39, no. 1, pp. 1–8, 2017.

[18] D. Sethi, P. Agrawal and V. Madaan, “X-Tumour: Fuzzy Rule-based Medical Expert System to Detect Tumours in Gynecology,” *International Journal of Control Theory and Applications*, vol. 9, no. 11, pp. 5073–5084, 2016.

[19] A. Jain and C. Gupta, “Fuzzy logic in recommender systems,” in *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications*. Cham: Springer, pp. 255–273, 2018.

[20] P. Kaur, P. Agrawal, S. K. Singh and L. Jain, “Fuzzy Rule-Based Students Performance Analysis Expert System,” in *Int. Conf. on Issues and Challenges in Intelligent Computing Techniques*, IEEEXplore, pp. 104–109, 2014.

[21] R. Kaur, V. Madaan and P. Agrawal, “Fuzzy expert system to calculate the strength / immunity of a human body,” *Indian Journal of Science and Technology*, vol. 9, no. 44, pp. 1–8, 2016.