Journal of Science and Technology
ISSN: 2456-5660 Volume 4, Issue 5, Sep-October 2019, PP39-48
www.jst.org.in

A Novel Cluster Setup for Fault Diagnosis of WSNS by Dag Scheduling and Log Files Using Internet of Things

Venkataramana K[1], Dr. Manoj Kumar[2]
[1](Reasearch Scholar, Department of CSE, SVU- Gajraula-UP)
[2](Associate Professor, Department of CSE, SVU- Gajraula-UP)
[1]Corresponding Author: kvrbook@gmail.com

Abstract: *A wireless sensors network (WSNs) contains thousands of sensor nodes which may lead to failures because of unattended deployments in it. Communication in sensor nodes will not happen due to these failures which produce in correct data due to its unstable nature of jumping from normal state to fault state results in data loss and sensor connectivity issues. Therefore it is necessary to take appropriate measures in advance to detect sustain network failures. To overcome this in our paper we proposed the cluster management setup in which the events of failures are detected, diagnosed and recovered by using the flow of DAG scheduler and LOG FILES.*
Keywords*: WSNs, Fault detection, Fault recovery, IOT, Fault diagnosis, DAG and LOG FILES,*

I. Introduction

A wireless sensor network (WSNs) comprises huge numbers of integrated sensor nodes and base station (BS). A Wireless Sensor Network (WSNs) comprises thousands of sensor nodes that may result in failure due to unsupervised deployments within. Interaction in sensor nodes would not occur due to each of these faults which lead to data failure and sensor connection issues in the correct data due to its unstable nature of jumping from normal to fault state. For this purpose, sufficient measures need to be taken in advanced to identify repeated network failures.[1].In order to overcome this in our paper, we suggested the cluster management system in which failure events are detected diagnosed and recovered using the DAG scheduler flow and log files.

Faults are unavoidable in wireless sensor networks owing to extreme environment and unsupervised deployment. Data communication and multiple network management cause sensor node deficiencies and so it is rare for sensor nodes to ignite their deficiencies and stop operation. [2] That can result in loss of connectivity and data. Hence it is appropriate to identify network faults in advance and to take appropriate steps to maintain network activity. Thus cluster-based data collection systems efficiently identify the faults, diagnose the faults and recover the faults.

The exact details of the single nodes are not identified till after deployment. Provided such a attributes, nodes were usually uniformly distributed in areas where access is hard or expensive. Each single node has only a fraction of the power, however the mix of thousands of nodes produces a great network able to identify, collecting, and monitoring environmental data. These may proceed to faults in the sensor network. Faults in the sensor network cannot be reported to wireless networks because of the following reasons [3]. Typical network protocols strive to achieve reliability point-to-point during which wireless sensor networks are far more worried with reliable detection of events. Faults happen more frequently in wireless sensor networks than those in typical networks, where it can be believed that client systems, servers, and routers function common. Hence, identifying failed nodes is important to ensure network connectivity and prevent network partitioning.

We planned to retain the management of the cluster in either the context of node and path failures. Within our system, the entire network is linked to the IOT connectivity sensors that monitor the progress of nodes and routes. IOT connectivity offers the centralized infrastructure and services that allow for cluster-wide synchronization. [4] This enables process synchronization by tracking a progress on servers that store information in local log files. A server has the ability to support a large cluster. If any of the nodes or routes service gets down, the working of this system is then our proposed scheme comes into the picture.

It enables us all to sign up as a service then again in our cluster setup, you can access information. Throughout this paper we study cellular architectural style and cluster-based to support network operation in case of failure caused by nodes drained from energy. In less than one-fourth of the time taken by the Gupta algorithm, the failure detection and recovery technique restores the cluster structure and is also shown to be 70 percent more energy efficient or the same. The cluster-based failure detection and recovery scheme is proving being an effective and practical solution for long and sustained operation of robust and extensible sensor network. Through cellular architecture the network is divided into such a computer cell grid to accomplish local fault detection and retrieval

with minimal energy consumption. Fault detection and retrieval in a distributed manner enables transmission of the failure report across cells. This algorithm is also being related to some existing work connected to it and has demonstrated to be more energy efficient. [2][3]

II. Related Work

To [5], Panda and Khilar recommended an altered self-fault diagnostic clustering algorithm on the three-sigma edit test to diagnose the soft faulty sensor nodes and time out processes for recognizing the hard faulty sensor nodes. The normalized median of absolute neighborhood deviation is measured and adjusted in a 3 sigma test edits for detecting the status of faults in nodes of sensors. This method will not be ideal for sensor node which is sparsely deployed. In DFI pseudo code which is proposed in the section notations, the notes and their definitions are used to create and evaluate.

A standard distributed-fault-detection (DFD) algorithm is proposed in Reference [6]. The DFD algorithm measures similarities of adjacent node sensed data simultaneously to evaluate the initial node state. The state and neighboring nodes check one another to decide whether a faulty nodes and the diagnosed results is being distributed to the nodes of neighbors, (DFD) approach must and should trigger the connecting nodes 3 times of that of nodes in neighbors evaluating the status of nodes, results in massive amounts of communication of data; (DLF) codes will absorb amounts of massive energy.

[7] Mei et al author suggested a method for using as an assistance of robots mobiles in the replacement of sensors for the nodes that failed. They are researching the algorithms to track, monitor failures of sensors and manage the movement of efficient energy. In [8], a protocol called as replacement for failures have been proposed in sensor networks of hybrid. To recover the faults and get the improvement in the coverage area of network and its connectivity the sensor mobiles are being used. The replacement initiates requests and sensors redundant will identified by the sensors of inadequacy.

In [9] DSC means dynamic-static-clustering protocol was introduced which was obtained from the protocol of LEACH. Distinction among dynamic-static-clustering and the protocol of LEACH is that (DSC) does not necessarily determine a CH at each round and does not do clustering. In certain terms, there is only one dynamic mode in the Leach protocol where the protocol of (DSC) will use mode of static and mode of dynamic. In the mode of dynamic all protocols are the same but in the static phase, they vary.

Imbalanced clustering with tolerance to faults is found in the PSO-UFC protocol. Such a protocol, described in [12],verifies the imbalanced clustering frameworks for balancing intra-cluster consumption and energy usage between major clusters of networks. In several hops, CH nodes all over the BS very quickly lose control. The PSOUFC protocol acknowledges additional CH called SCH for network CH nodes. As a result such a protocol always repairs and maintains the optimum level of network connectivity.

The device laid down in [13]was called ECraft. This system has increased fault tolerance capabilities across nodes and touch level. At the same time, ECraft used all 3 methods of self identification, group retrieval, and hierarchical retrieval to detect faults. In the Fault Recovery phase this framework has only implemented Deny of value for NCH nodes. In addition, using the policies of Hierarchical Election, Self-Election and Team Election, the selection of a current cluster leader during the Recovery period was applied in a coordinated and synchronized way. Finally, pre-copy and remote execution methods were used in the recovery phase of Service Distribution.
Objectives identified and proposed in our paper are.

- The rest of the paper contains section III is the proposed scheme, we designed an efficient novel Cluster set of network model for identifying the faults in sensor nodes.
- Section IV is all about the scheduling of F-DDR with a DAG scheduler and LOG FILES with a working and has found the probability of it.
- Section IV is the conclusion.

III. Proposed Scheme

3.1 Proposed Network Model For Fault Identification

Suggested Architecture operates on the Client –Server architecture where clients are our cluster management setup system nodes and servers are sensor nodes that aggregate the data from CHs to base station. The figure below shows the relation between the sensor node servers and their cluster setup clients. The cluster setup and the sensor nodes are connected to a coordinator (CHs) which gives our client cluster the status of all sensor data. The proposed Framework comprises components

1. **Client node** is used for accessing information from the server sensor nodes in our distributed system of cluster. Messages are being sent to the server sensor nodes (cluster head (CHs) and letting the server know that client is a-live and if response is not from the connected servers of (CHS) again the clients send backs a message to connect automatically to other servers of (CHs).
2. The **server sensor nodes** (CHs) send the client an acknowledgment to notify that the server sensor node is alive, and provide all cluster client services to access all sensing data.
3. **Leader** here acts as CH. If either of the server nodes fail this server node will automatically recover.
4. **Coordinator** is a node in server coordinates the instructions which are given by the leader represented as L (CH).

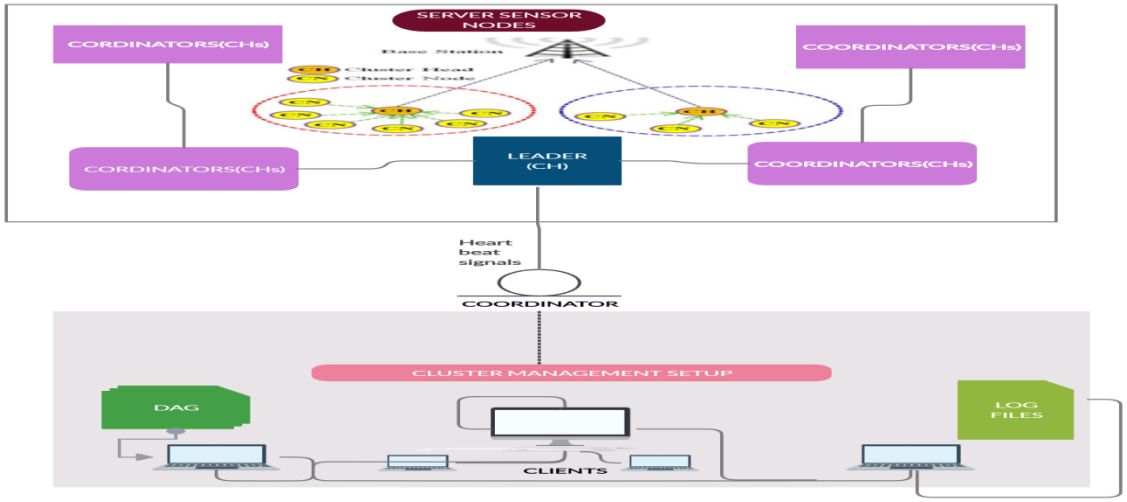
3.2 Working of Proposed Architecture

The very first thing that happens if a collection of sensor servers starts is the clients going to wait to connect to the servers. Here server acts as sensors and clients act as the setup for our cluster management. Then our cluster configuration group's clients must connect to one of the sensor nodes that act as a leader node. Here CH serves as the so-called coordinator leader node. Once the configuration of our client cluster is connected to the CH leader node, the node will then assign the client a session ID and send that client an acknowledgement. If a client gets zero acknowledgment from either the node, it will send back the request to the next node on the other CH server and try to communicate.

Our architecture selects someone else as leadership role (CHs), while others act as coordinators. Eventually, to detect the faults that happen, the client can perform functions such as reading, writing, and store the data track and status as per. Proposed system benefits are robust as it continues to perform even if more than one node fails, it operates with a ratio of 10:1 in cases where' faults' are more likely to occur, efficiency can be improved by more computer deployment. The proposed architecture provides us with a graphical representation of the detection of faults, the diagnosis of faults and the recovery of faults which is done using a DAG scheduler.

- **Fault detection:**capable of detecting any node which allows the cluster for storing the information which is to date of each node and detecting unwanted nodes that lead to security breaches. This can be achieved with a DAG scheduler and data stored in log files.
- **Fault diagnosis**: it handles cluster in a way of each node's status is preserved in time of real, fewer leaving chances of error and uncertainty. Fault diagnosis is performed with a unique identification which attaches to each sensor node.
- **Fault recovery:** Failure Recovery locks log files into our cluster setup. While any changes occur in the data of the sensor, we can track each and every modification of the node from which any faults will occur then we can recover those faults.

Figure 1 Proposed Cluster Setup For Fault Daigonis In WSNs



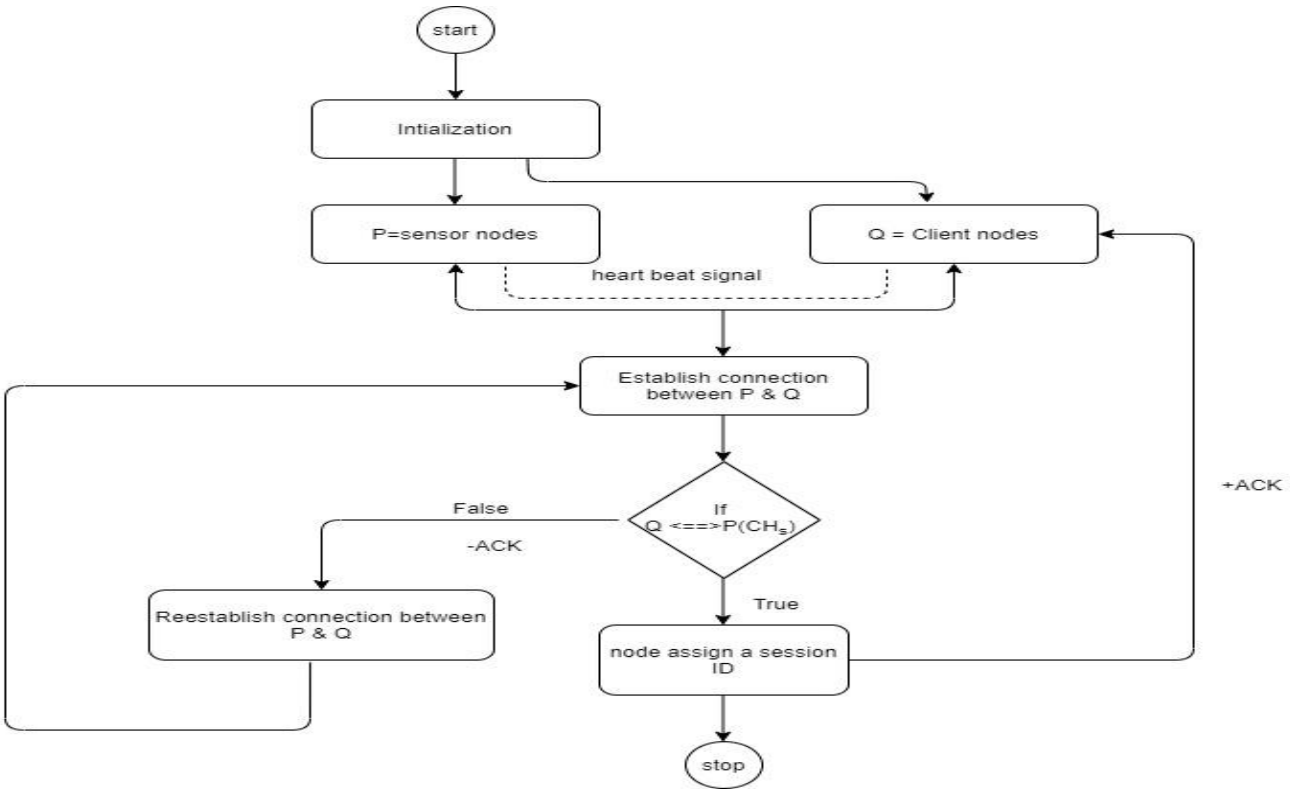
3.3 Algorithm For Connectivity Of Cluster Setup And Sensor Nodes

1. Start
2. Initialization let say P= server's acts as sensors; Q=clients act as our cluster management setup; coordinator=CH
3. Establish connectivity with P and Q
4. Case1:Positive acknowledgment
5. IF(Q $\Longleftrightarrow$ P(2CHs))
6. NODE ASSIGNS A SESSION ID
7. ACK→Q
8. ELSE
9. Case2:Negative acknowledgment
10. Re-establish connectivity again with P and Q
11. (Q $\Longleftrightarrow$ send message to another node in P(2CHs))
12. ACK→Q
13. connection is establishes
14. sends heartbeat signals at regular intervals
15. our required operations can be performed
16. stop

3.4 Flow Of Connectivity Of Cluster Setup And Sensor Nodes

In the below flow, initially a connection is needed to establish between client and sever nodes. For this purpose the client node which is represented as Q sends a message to the elected cluster head (CH) of server nodes say as P.P and Q establishes the connection, for every interval a heart beat signal is releases which is an indicator that connection is established. If the connection is not established then again it sends a message to other CH and tries to establish the connection until Q receives a positive acknowledgment from P.

Figure 2: P and Q Connectivity Flow Chart



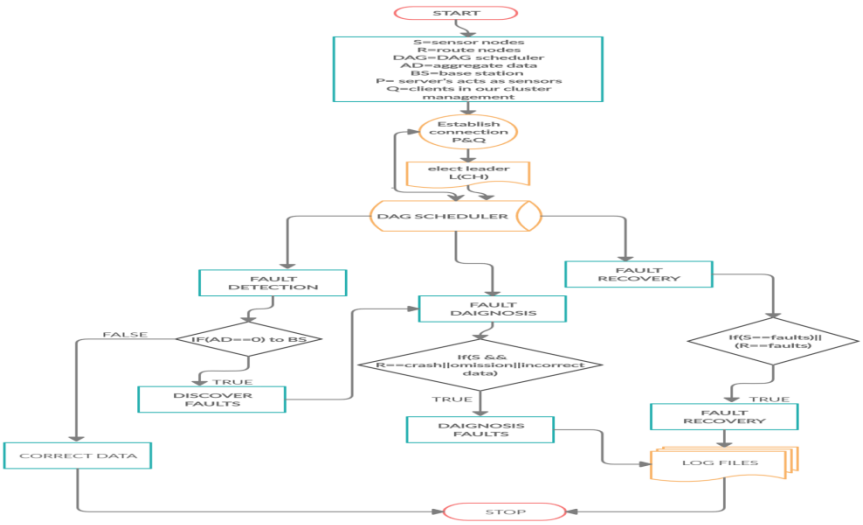
3.5 Algorithm For Finding Faults Using Our Cluster Management Set Up

1. start
2. sensor nodes say S; route nodes say R; DAG scheduler say DAG; aggregate data=AD; base station BS; P= server's acts as sensors; Q=clients act as our cluster management setup
3. establish connectivity between P&&Q
4. Elect leader to cluster head(CH) say as L(CH)
5. L(CH) ~~<-->~~ DAG
6. Fault detection phase
7. If(AD==0)to BS
8. Discover faults; perform step 13
9. Else
10. Correct data is transferred
11. End if
12. Fault diagnosis phase
13. If(S && R==crash||omission||incorrect data)
14. Go to DAG
15. Diagnosis the faults
16. If(S==faults)|| (R==faults)
17. Fault recovery phase
18. Track path of S and R in log files
19. Recover the faults
20. Stop

3.6 Flows For Finding Faults Using Our Cluster Management Set Up

Initially the link between server side sensor nodes and client side cluster setup is being built. Then a leader is chosen to say cluster head of sensor nodes as L (CH) which is connected to the DAG scheduler where the faults are found. DAG scheduler is scheduled in 3 phases such as phase of identification of faults, phase of fault diagnosis and phase of recovery of faults referred to as F-DDR. Faults are identified as if the aggregated data from cluster heads to BS is zero in the fault detection phase, and then there are faults that need to be identified and diagnosed. So the next step is fault diagnosis where faults for sensor nodes will occur or route path nodes will be diagnosed with DAG scheduler support. Recovery of faults in this process is achieved by log files. Thus DAG scheduler plays a major role in identifying all the faults that make our system faults free.

Figure 3: Flow Chart For Finding F-Ddr Using Dag And Log Files



IV. Working of Dag Scheduler For F-DDR

DAG helps us achieve the fault tolerance by monitoring the faults in our cluster setup. The WSNs node faults generally occur due to failure to send adequate data to the base station due to misbehaving security breaches; failure of modules such as communication and sensing module etc. You can track all of these faults using a DAG scheduler. It is a set of vertices and edges where vertices represent nodes and edges of CHs represent the connectivity to specific sensor nodes that our proposed scheme applies to operations on.

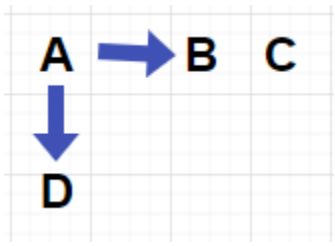
In DAG scheduler, each edge guides in the sequence from earlier to later; when any data connection is completed, CREATED DAG submits the graph to the DAG scheduler, which divides the graph into stages of the task from CHs to base station. The benefits of DAG connectivity are nodes that can recover using the Directed Acyclic Graph, we have multiple graph representation rates that will make it easy and scalable for us to map faults, allow us to achieve fault tolerance, recover our data on losses and achieve global optimization.

Let's take nodes say n1, n2, n3.When a node collapses in either the center of a process, say n3 based on n2 operation that in effect is n1. A cluster operator figures out that node becomes killed and selects another node to continue operation such node must run on the specific operation position sequence it has to perform (n1->n2->n3). There will now be no loss of data.

4.1 Probability of F-DDR Based on Dag Schedular

Relationship between DAGs and probability distributions is done with a few examples how exactly it happens. DAGs encodes assumptions about the dependencies between sensor nodes. DAGs will tell us which nodes are independent from each other? Which nodes are conditionally independent from each other? I.e. the ways that we can factor and simplify the joint distribution

Example 1



A DAG involving nodes A, B, C and D encodes assumptions about the joint distribution P (A, B, C, and D)

This DAG implies

$$P(C|A,B,D) = P(C) - - - - - -> (1)$$

I.e. C is independent of all variables

$$P(B|A,C,D) = P(B|A) - - -\rightarrow (2)$$

I.e. B ||D, C|A

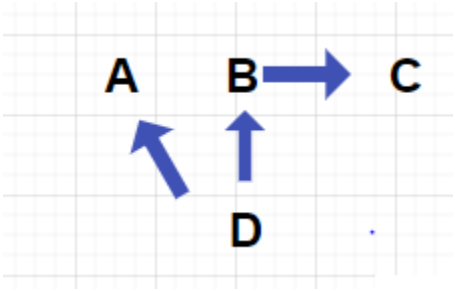
$$P(B|D) \neq P(B) - - - - -\rightarrow (3)$$

B and D are marginally dependent

$$P(D|A,B,C) = P(D|A) - - - -(4)$$

Example 2

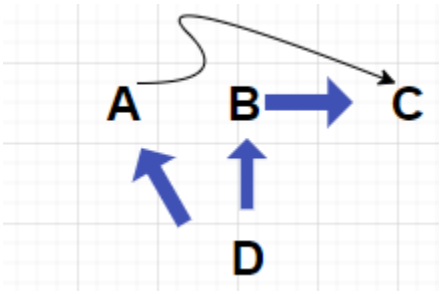
This DAG implies



$$P(A|B,C,D) = P(A|D) - - - - - -> (5) A||B,C|D$$

$$P(D|A,B,C) = P(D|A,B) - - - \rightarrow (6) D||C|B$$

EXAMPLE 3



This DAG implies

$$P(A|B,C,D) = P(A|C,D) - - - - - -> (7) A||B,C|D$$

$$P(D|A,B,C) = P(D|A,B) - - - \rightarrow (8) D||C|A,B$$

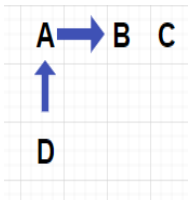
Relationship Between Dags and Probability Distributions

Decomposition Of Joint Distribution

We decompose the joint distribution by sequential conditioning only on sets of parents.

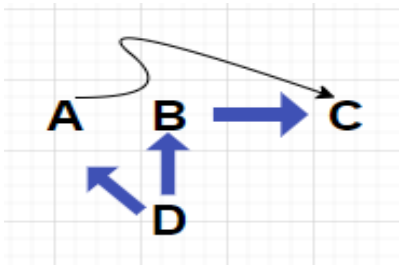
Start with nodes with no parents. Proceed down the descendent line, always conditioning on parents.

Decomposition example 1



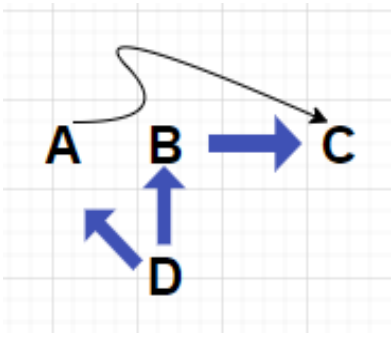
$$P(A,B,C,D) = P(C)P(D)P(A|D)P(B|A) - - - -(9)$$

Decomposition example 2



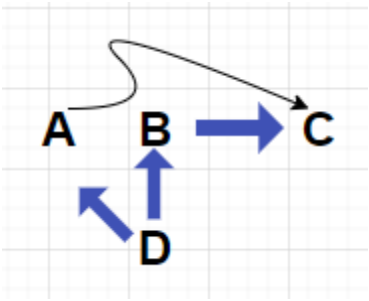
$$P(A,B,C,D) = P(D)P(A|D)P(B|D)P(C|B) - - - -(10)$$

Decomposition example 3



$$P(A,B,C,D) = P(D)P(A|D)P(B|D)P(C|A,B) - - - -(11)$$

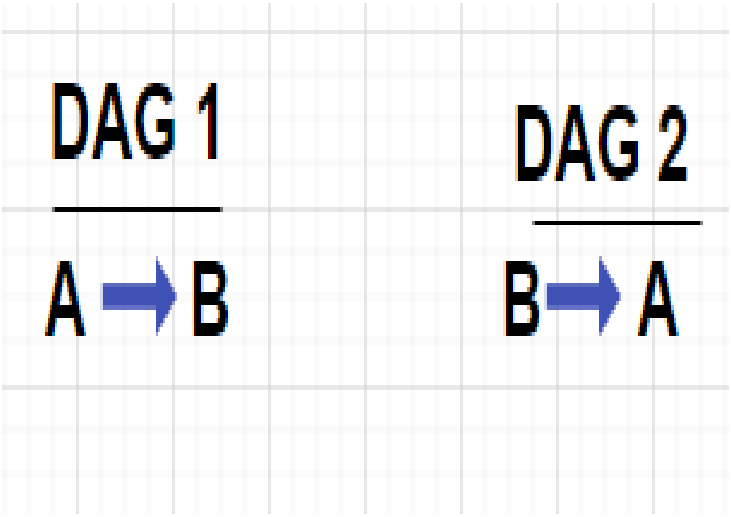
Compatibility Between Dags And Distributions
This DAG Admits Its Factorization



$$P(A,B,C,D) = P(D)P(A|D)P(B|D)P(C|A,B) - - - -(12)$$

This probability function and this DAG are compatible

Compatibility Between Dags And Distributions
DAGs that are compatible with a particular probability function are not necessarily unique.
Simple example



These both convey that A and B are dependent

i.e...$P(A,B) \neq P(A)P(B) - - - (13)$

V. Conclusion

In our paper a cluster management setup is proposed to achieve F-DDR which finds the faults in the sensors of WSNs by using directed acyclic graph (DAG). Therefore, it becomes necessary for nodes in sensors to detect and diagnosis the readings of faults with our proposed set up to recover the faults. Hence in our cluster setup we can achieve the F-DDR by means of DAG scheduler for detection and diagnosis of faults and for recovery of phases by means of LOG FILES. Our scheme gives an efficient, scalable, fault tolerant solution.

References

1. G.Y. Durga Devi (2013),"Clustering Algorithms In Wireless Sensor Networks-A Survey, ISSN (Online): 2347-2820, Volume -1, Issue-2
2. J.Neuzil,O.Kreibich,andR.Smid,"Adistributedfaultdetectionsystem based on iwsn for machine condition monitoring," IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1118–1123, May 2014.
3. I. Bekmezci and F. Alagoz, "Energy efficient, delay sensitive, fault tolerant wireless sensor network for military monitoring," in Sensors Applications Symposium, 2008. SAS. IEEE, 2008, pp. 172–177.
4. H. Xu, R. Zhang, C. Lin, and Y. Ma, "Novel approach of fault diagnosis in wireless sensor networks node based on rough set and neural network model," International Journal of Future Generation Communication and Networking, vol. 9, no. 4, pp. 1–16, 2016.
5. S.Rani,S.H.Ahmed,R.Talwar,andJ.Malhotra,"Cansensors collect big data? An energy-efficient big data gathering algorithm for a WSN,"IEEE Transactions on Industrial In formatics, vol.13,no.4,pp.1961–1968, 2017.
6. Kaiwartya, O., Abdullah, A. H., Cao, Y., Lloret, J., Kumar, S., Shah, R. R., et al. (2018). Virtualization in wireless sensor networks: Fault tolerant embedding for internet of things. IEEE Internet Things, 5, 571–580
7. Ding, S. X., Zhang, P., Yin, S., & Ding, E. L. (2013). An integrated design framework of fault-tolerant wireless networked control systems for industrial automatic control applications. IEEE Transactions on Industrial Informatics, 9(1), 462–471.
8. MEl Y, XIAN c., DAS S., HU yc., LU Y.H.: 'Repairing sensor networks using mobile robots'. Proc. ICDCS In!. Workshop on Wireless Ad Hoc and Sensor Networks, Lisboa, Portugal, July 2006
9. M. Panda and P. Khilar, "Distributed self fault diagnosis algorithm for large scale wireless sensor networks using modified three sigmaedittest," Ad Hoc Networks, 2014.
10. Cheraghlou, M. N., & Haghparast, M. (2014). A novel fault-tolerant leach clustering protocol for wireless sensor networks. Journal of Circuits, Systems, and Computers, 23, 1450041
11. Rajeswari, K., & Neduncheliyan, S. (2017). Genetic algorithm based fault tolerant clustering in wireless sensor network. IET Communications, 11(12), 1927–1932.
12. Kaur, T., & Kumar, D. (2018). Particle swarm optimization-based unequal and fault tolerant clustering protocol for wireless sensor networks. IEEE Sensors Journal, 18(11), 4614–4622.
13. Cheraghlou, M. N., Khadem-Zadeh, A., & Haghparast, M. (2017). Increasing lifetime and fault tolerance capability in wireless sensor networks by providing a novel management framework. Wireless Personal Communications, 92(2), 603–622.