Data Recovery Approach for Fault-Tolerant IoT Node

Perigisetty Vedavalli, Deepak Ch
School of Electronics and
Communication Engineering
VIT-AP University
Amaravathi, Andhrapradesh, India 522237

Abstract—**Internet of Things (IoT) has a wide range of applications in many sectors like industries, health care, homes, militarily, and agriculture. Especially IoT-based safety and critical applications must be more securable and reliable. Such type of applications needs to be operated continuously even in the presence of errors and faults. In safety and critical IoT applications maintaining data reliability and security is the critical task. IoT suffers from node failures due to limited resources and the nature of deployment which results in data loss consequently. This paper proposes a Data Recovery Approach for Fault Tolerant (DRAFT) IoT node algorithm, which is fully distributed, data replication and recovery implemented through redundant local database storage of other nodes in the network. DRAFT ensures high data availability even in the presence of node failures to preserve the data. When an IoT node fails in any cluster in the network data can be retrieved through redundant storage with the help of neighbor nodes in the cluster. The proposed algorithm is simulated for 100-150 IoT nodes which enhances 5% of network lifetime, and throughput. The performance metrics such as Mean Time to Data Loss (MTTDL), throughput, Network lifetime, and reliability are computed and results are found to be improved.**

Keywords—Internet of things; data recovery; RAID; node failures; reliability; network lifetime

I. Introduction

IoT provides an integration of multiple controllers, IoT nodes, servers, and gateways which contains embedded technologies to be logically connected and enables them to sense and interact to the real world and also among themselves. It is attainable for them to gather data from a wide range of existing structures. The accuracy of the IoT network is diminished, when the transmission data is faulty which leads to cause for inappropriate actions. So, it is critical to enhancing the ability of the IoT network to detect and recover the faulty node's data. Difficulties of detecting incorrect data and the quality of data have been studied extensively [1]. Fault tolerance is an important aspect for ensuring the high reliability and availability of the IoT network. Due to resource-constrained devices, there may be a lot of chances to occur failures in the network. Hence traditional networking technologies cannot handle IoT requirements effectively [2]. According to the survey, 48% of IoT projects may fail due to data failures and security. Data failures include missing files, corrupt files, and data blocks, and inconsistent files [3].

Capturing the data quality levels is more effective to estimate the device's quality and their produced data. Data quality estimation mechanism depends on three stages[4], data reliability, device availability, and overall quality of data.Sensors are small devices with limited resources like memory space, battery, and processing power. These resource-constrained nodes pose multiple challenges for network designers' accurate usage of scarce resources. In certain times IoT applications need to be deployed in harsh environments [5]. In such IoT applications, nodes are prone to failures due to various reasons, such as hardware failure, battery depletion, and external events. Complex IoT applications with the help of various techniques require effective data management [6]. IoT is a complex heterogeneous network, maintaining high reliability is one of the major concerns. IoT networks should be more reliable for safety and critical applications, with the help of heuristic binary decision diagrams can able to access the link failures that occur due to data loss in community structures [7].

IoT applications help the industries to bring a competitive edge on their competitors. Even due to sharing data, security, device faults, and data manipulation between the various smart devices which becomes a serious concern to many industries, these interrupt the workflow of industries. IoT network comprises many sensing nodes, so the network needs to collect and process an enormous amount of external environment data. Enhancing the fault tolerance and reliability can be achieved by adding redundant bits to the original data at the information level is necessary for an IoT network. By employing the Reduced Variable Neighborhood Search (RVNS) algorithm the IoT network can enhance the processing speed and reliable transmission of data [8]. Generally, sensor data validation includes mainly two steps, those are data faults detection and data reconciliation. There is no perfect tool or method for this process.

In [9] various faulty data detection and correction methods and tools are discussed. In an IoT network, each sensor node works with a limited power source and when the sensor stops working the network cannot process data that they received this may affect the prediction of network health and reliability which leads to network failure. When this situation happens RAID structures are very useful for maintaining the redundant copies of data. This paper focuses on data management with the help of a particular RAID-like technique in the cluster for achieving fault tolerance with additional communication costs.

The remaining paper is organized as Section 2 recent data replication and recovery methods in IoT, Section 3 is about the implementation of a DRAFT algorithm in the IoT network. Section 4 is discussed about performance evaluation for DRAFT Algorithm and finally, section 5 draws conclusion and possible future directions of work.

II. Related Work

IoT requires efficient data collection, generation, and presentation through wireless sensor nodes. Due to limited energy resources, sensor nodes are unreliable, which may lead to the loss of valuable data. In IoT, data replication is a promising method for data management. In [10] data replication algorithms for IoT nodes classification, analysis, and comparison are presented. Data replication techniques include query balancing, data availability, system robustness, and data retrieval Resilient data-centric storage algorithm is utilized by [11] to illustrate the tiny database systems for easy data retrieving. It is specially designed for low-powered IoT sensor nodes. A Distributed Hash Table(DHT) is used for storing the redundant data into the node.

The redundant storage of data assures the information available in case of source failure. A greedy replication-based distributed storage algorithm is proposed in [12], if a node in the network fails then the data can be retrieved through neighbor donor nodes. IoT sensor nodes data is stored in distributed mini data centers in a decentralized manner cloud rather than single cloud storage. In those data storage areas, each IoT data item has predetermined redundant data copies. This problem has been formulated with the help of a complex-linear programming model and heuristic algorithms are proposed in [13]. These algorithms are helping to improve the latency of reading and writing operations.

A multi-dimensional data storage algorithm is implemented within a single node [14], which can create to handle querying, indexing, storing, and ingestion of huge amounts of data. A distributed MDDS offers high ingestion rates, fault tolerance, and horizontal scaling when compared to Relational database management systems (RDBMS). To assure high data availability in the IoT network during the node failure distributed hop by hop data replication (DRAW) technique is proposed in [15]. This algorithm helps to identify the best replica node for maintaining a redundant copy. For selecting the replica node this technique applies a series of conditions like availability of the memory in the device, the number of hops, degree of replication, previous replicas of the data items, and common neighbors of the devices.

Data availability during the failures can be achieved through data replication algorithms. In [16] bridged replication control algorithm (BRC) is proposed for smart logistics. BRC creates temporary replication access when the link failure occurred through the bridge token. BRC provides efficient data management for smart networks. By maintaining redundant data components in multiple storage locations can achieve high fault tolerance. An adaptive data replication algorithm is proposed [17], and incorporated at the gateway level.

IoT network is deployed with many sensor nodes, in this context energy of the sensor nodes is depleted during data transmission. An adjustable data replication algorithm based on virtual grid technology is implemented in [18]. This scheme helps to enhance the lifetime and performance of the sensor nodes. A dynamic sink node will determine the communication link depending on the selected beacon and continuously develop a replica node across the query node to create a balance between the rate of energy consumption and the overhead in the network. Data recovery is one of the essential features of an IoT network. These networks may face some issues due to sensing and connection errors which result in incorrectly received data [19]. By incorporating probability matrix factorization at the cluster level can recover the missing data through neighbor nodes in the cluster.

TABLE I. Comparison of Various Fault Data Recovery Approaches and Replication Techniques

Author	Method	Single/ Multi Node point Failure	Fault recovery and prediction	Replica Distribution	No.of Nodes
Shong[9]	Edisense	Multi point failure	Fault handling Mechanism	Locality based	10-20 Nodes
Qaim[13]	Distributed hop by hop replication algorithm	Single point failure	Data errors handling	Connectivity energy based	50-70 Nodes
Juan[14]	Low complexity greedy algorithm	Single point failure	Data errors handling	Memory based	10-50 Nodes
Qaim[15]	Adjustable data replication algorithm	Single point failure	Data errors handling	Uniform based	100 Nodes
DRAFT algorithm	RAID based replication algorithm	Single point failure	Data errors and fault handling	Probabilistic and memory based	100-150 Nodes

In [20] a convolution neural network has been utilized for generating data recovery algorithms. For restoring the data which mainly includes two steps, all sensed collected for the training process to the networks and data recovery has been initiated with the help of a trained generator.An redundant residue number system algorithm [21], Max-flow algorithm fault tolerant [22], Device pairing algorithm [23], Finding Least connected points algorithm [24], Least connected neighbour algorithm [25] are the few fault-tolerant algorithms have been studied. The above-listed literature algorithms in Table I were presented with a minimum of 10 to a maximum of 100 nodes. This work was analyzed properly to extend the existing algorithms works towards 150 IoT nodes with network lifetime, MTTDL, Reliability, and throughput of the network.

III. Implementation of a DRAFT Algorithm in the IoT Network

Safety and critical industrial IoT applications such as IoT based nuclear radiation monitoring systems require more data availability and high fault tolerance during data loss. Thus protecting sensed data through redundant storage can significantly improve the system performance. Let us consider ‘n’ clusters with ‘N’ static nodes are randomly deployed in a sensing field. Each node has a unique ID, ‘i’ where i=1,2,3,. . . ., P. Each node ‘i’ has fixed memory space ‘k’, to hold data and parity items. The memory space is partitioned into two units based on the RAID5 structure, one is for storing data and another unit is for parity storage. RAID is a well-proportioned technology that creates improved storage reliability and functions by block-level striping with parity in the node storage area. At

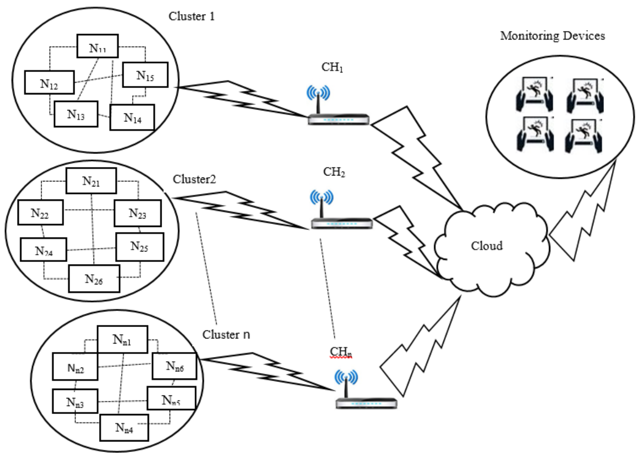

Fig. 1. Cluster Based Multi-Node IoT Architecture.

regular intervals, all the nodes detect environmental factors and produce sensing data D_i. The effective capacity for storing the sensing information is (k-1)/k of its total storage ‘k’. 1/k storage space is utilized for storing redundant information. Each node preserves a Direct Neighbor Node(DNN) attribute list $'N'_{Att}$. Every sensor node in the network can produce data and updates in the data unit as well as parity unit, with the help of all DNN in the cluster, and the node failure is denoted as N_F. To implement the DRAFT algorithm, a few assumptions have been made regarding the cluster-based multi-node IoT architecture as follows

- Assume that each cluster has at least five nodes.
- All nodes are having identical significance and storage space.
- All the nodes are having similar computation resources and initial energy supply.

To follow these steps the architecture is built as shown in Fig. 1. The basic architecture of IoT nodes consists of five elements they are controller, sensing element, battery, local storage, and network connectivity. The local storage of the node is partitioned into two parts. A DRAFT algorithm is invoked at each node and its storage unit is partitioned into two units based RAID5 mechanism. The inputs of the algorithm are no. of clusters ‘n’, storage space ‘k’, Data items D_i , 1/k parity unit, (k-1)/k data unit space, DNN attribute table N_{Att}. As the outputs of the algorithm create a node ID, and generate the parity based on DNN, data recovery from the DNN, MTTDL, throughput, network lifetime, and reliability. For every round, if any node data loss occurs can be retrieved through the DNN of that particular node in the cluster. A node collects new data samples continuously, and that data only updates its D_i section, but it also transmits a copy of the newly detected data to all of its direct neighbors, allowing them to update their Pi sections by computing the parity from the received data. The procedural steps for DRAFT Algorithm in IoT network as follows:

DRAFT Algorithm

```
Step1 Input declaration
Input:Number of clusters n, Data item Di, storage space k, DNN attribute table $N_{Att}$
Step2 Output declaration
Output:Parity generation, data recovery if any data loss occurs,MTTDL,Network Lifetime, throughput,and reliability.
Step3 Node structure
Algorithm defnode (node id, cluster id, Data Di, parity Pi)
begin
$C_{id}$ = Cluster id
$N_{id}$ = Node id
$D_i$ = Data of ith node
$P_i$ = Parity of ith node
end
Step4 Storage unit structure
Algorithm DRAFT (n, k, DNN)
def node
Data unit = (k-1)/k
Parity unit = 1/k
update $D_i$, $N_{att}$, $P_i$
Step5 Initialize the first round
Node i transmitted data to cluster head i
if ($D_i$ of $N_i$ = $D_i$ of $C_i$)
transmit ACK
{
initialize next round
}
Else (error detection=true)
Transmits NACK
{
For each node in cluster A data recovery
        $DL_{iA} = \sum_{j=1}^{n} D_{jA}$
$P_i$ = XOR(DNN data of $i^th)$ node)
Data Recovered successfully
transmit ACK
}
Step6 Network lifetime extraction
Network lifetime
{
$P_v$= $(1-p)^(N_d)$
}
Step7 MTTDL
{
MTTDL = $\int_0^\infty$ P($S_0$(t))+P($S_1$(t))+P($S_2$(t))dt
        $MTTDL = \frac{H(x)}{V(x)}$
    Where as
H(x) = $(\mu_D + k\lambda_0 + (k-1)\lambda_1)(\alpha_1 + (k-1)\beta_1) + (k\lambda_0 + (k-1)\lambda_1)(\mu_D + \lambda_2 + (k-1)\lambda_1)$
$V(x) = (k\lambda_0(k-1)\lambda_1(\alpha_1+\lambda_2+(\mu_D(k-1)\lambda_1)(k-1)(\lambda_1+\beta_1)$
}
if $\mu_D --> \infty$
{
$MTTDL = \frac{\alpha_1+k\lambda_0+(k-1)(\lambda_1+\beta_1)}{k\lambda_0(k-1)(\lambda_1+\beta_1)}$
}
if $\mu_D --> 0$
```

```
{
$MTTDL = \frac{1}{k\lambda_0} + \frac{1}{(k-1)\lambda_1}$
}
}
Step8 Reliability extraction
{
$P(S_0(t)) + P(S_1(t)) + P(S_2(t)) + P(S_F(t)) = 1$
For each cluster A in N {
    $\frac{\partial P(S_0(t))}{\partial t} = - k\lambda_0 P(S_0(t)) + \alpha_1 P(S_2(t))$
    $\frac{\partial P(S_1(t))}{\partial t} = k\lambda_0 P(S_0(t)) - (\mu_D + (k-1)\lambda_1 P_1(t) + \lambda_2 P(S_2(t))$
    $\frac{\partial P(S_2(t))}{\partial t} = \mu_D P(S_1(t)) - (\alpha_1 + \lambda_2 + (k-1)(\lambda_1 + \beta_1))P(S_2(t))$
    $\frac{\partial P(S_F(t))}{\partial t} = (k-1)\lambda_1 P(S_1(t)) + (k-1)(\lambda_1 + \beta_1)P(S_2(t))$
}
Step9 Throughput
{
if $D_i$ of $N_i = D_R$
{
Successful recovery
}
else
{
Unrecoverable data
}
Repeat
End
```

Parity is computed by XORing all the inputs in a bit-wise manner. If there are more than one boolean inputs, XOR returns true when the two inputs are different. A parity scheme is one of the common approaches for error detection. Cluster1 is grouped with five nodes each node is connected with three nodes. If any node failure causes the neighbor nodes will help to retrieve the data of that failed node by using redundant information. Data storage in D_i and P_i and recovery is represented in Fig. 2 topology. Here defining the following relations for nodes with their corresponding neighbors,

$$P_1 = D_1 \oplus D_2 \oplus D_3 \oplus D_5 \tag{1}$$

$$P_2 = D_1 \oplus D_4 \oplus D_3 \oplus D_2 \tag{2}$$

$$P_3 = D_1 \oplus D_4 \oplus D_5 \oplus D_3 \tag{3}$$

$$P_4 = D_2 \oplus D_3 \oplus D_5 \oplus D_4 \tag{4}$$

$$P_5 = D_1 \oplus D_4 \oplus D_3 \oplus D_5 \tag{5}$$

By using the above equations 1 to 5, the pi will store the information in cluster 1, similarly, all the clusters in the network the parity unit will update based on the DNN of the particular node in the network. Di unit will update the sensing data into it.

Using parity can easily rebuild the lost data inputs by conducting XOR of all the remaining values and former output values. Assume that node2 fails the data recovery of the node2 will get with the help of redundant information stored and its corresponding neighbors, Identified that if D2 and D5 at round1 and round2 instances then data recovery has been done with the help of all DNN and parity of the failed node in that cluster which is represented in Table II. Hence the data is proved that fault recovery has been made using a DRAFT algorithm. IoT network is grouped with 'n' no. of clusters, each cluster consists of $n \geq 5$ nodes. Each node has four states they are normal state, degraded state, recovery state, and failed state. The reliability of the network and MTTDL of the node has been derived with the help of the following state diagram Fig. 3.

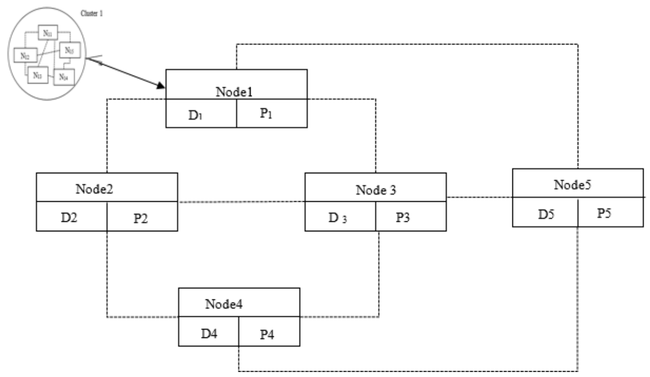

Fig. 2. Representation of Cluster1 Data Storing and Communication Process.

TABLE II. DATA RECOVERY PATTERNS OF CLUSTER1 FOR ROUND1 AND ROUND2

Cluster1				
Transmission Data	Parity	Receiving Data	Faulty Data Identification	Data Recovery
D1 = 100	P1 = 111	100	ACK	No error
D2 = 101	P2 = 101	111	NACK	error occured, Recovery intiated, D2= 101
D3 = 111	P3 = 001	111	ACK	No error
D4 = 011	P4 = 000	011	ACK	No error
D5 = 001	P5 = 001	001	ACK	No error
D1 = 101	P1 = 100	101	ACK	No error
D2 = 101	P2 = 011	101	ACK	No error
D3 = 111	P3 = 101	111	ACK	No error
D4 = 100	P4 = 101	100	ACK	No error
D5 = 011	P5 = 101	001	NACK	error occured, Recovery intiated, D5= 011

Let us consider the nonzero time of node replacement, the failure rate of a node is normal in degraded and rebuild states. S_0 is the normal state, in this state all the nodes in the cluster are operable and data in every node is available. From this state, the node can pass to state S_1 with the rate of k0(Failure of any node).
S_1 is a degraded state, in this state one of the nodes in the cluster has been failed and waiting for a replacement and the remaining k-1 nodes are operable and data can be available. From this state, the node can move to either state F with the failure rate $(k-1)\lambda_0$ (failure of another operable node) or S_2 with the repair rate of $_D$.
S_2 is the recovery state, in this state the failed node is replaced and the recovery process has been started, the remaining k-1

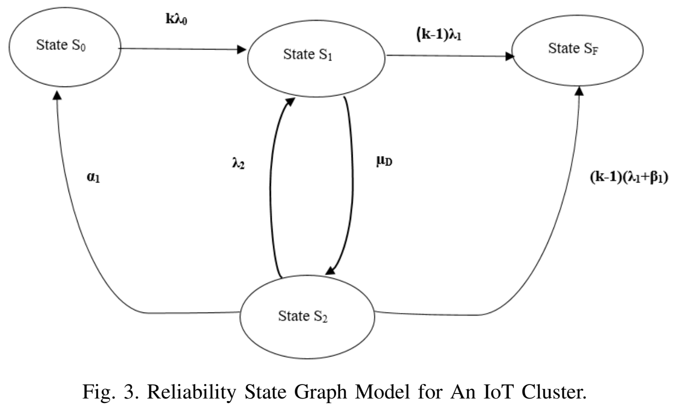

Fig. 3. Reliability State Graph Model for An IoT Cluster.

nodes are operable and data is available. From this state, the node can move to either state0 with the rate of α_1 (successfully data recovery completed) or to S_1 with the failure rate λ_2 (failure of the node during the data recovery process) or to S_F with the rate of (k-1)λ_1 (failure of one of the operable nodes) or with the rate of (k-1) β_1 (read error on one of the operable nodes during data recovery process).
S_F is the failed state data of the node that are non-recoverable and unavailable. Let us present the state diagram based on the above transitions.

Where as

- λ_0 is the failure rate of the node in a cluster.
- λ_1 is the failure rate of the node in case of data unavailability of the node which is operable.
- λ_2 is the failure rate of the replaced node during the data recovery.
- μ_D repair rate of the faulty node.
- α_1 is the success rate of data recovery.
- β_1 rate of reading errors on the operable nodes during the data recovery.

The above state diagram is solved with the help of Kolmogorov-Chapman differential equations which are analyzed as follows:
Initially the probabilities of state S_0 is $P(S_0(t)) = 1$, the remaining states probability which is equals to zero, hence $P(S_1(t)) = P(S_2(t)) = P(S_F(t)) = 0$.

$$P(S_0(t)) + P(S_1(t)) + P(S_2(t)) + P(S_F(t)) = 1 \tag{6}$$

$$\frac{\partial P(S_0(t))}{\partial t} = -k\lambda_0 P(S_0(t)) + \alpha_1 P(S_2(t)) \tag{7}$$

$$\frac{\partial P(S_1(t))}{\partial t} = k\lambda_0 P(S_0(t)) - (\mu_D + (k-1)\lambda_1 P_1(t) + \lambda_2 P(S_2(t)) \tag{8}$$

$$\frac{\partial P(S_2(t))}{\partial t} = \mu_D P(S_1(t)) - (\alpha_1 + \lambda_2 + (k-1)(\lambda_1 + \beta_1))P(S_2(t)) \tag{9}$$

$$\frac{\partial P(S_F(t))}{\partial t} = (k-1)\lambda_1 P(S_1(t)) + (k-1)(\lambda_1 + \beta_1)P(S_2(t)) \tag{10}$$

The reliability shows that only from state 0-2, the cluster will be in operational mode, and the data is also available. To derive the formula for mean time to data loss (MTTDL) for the cluster, considering the cluster will be staying at state0 – state2 and taking into that initial state of the cluster is state0 for calculating the MTTDL as follows:

$$MTTDL = \int_0^\infty P(S_0(t)) + P(S_1(t)) + P(S_2(t))\,dt \tag{11}$$

By substituting the above values 7,8,9 and 10 in 11 we get,

$$MTTDL = \frac{H(x)}{V(x)} \tag{12}$$

Where as
$H(x) = (\mu_D + k\lambda_0 + (k-1)\lambda_1)(\alpha_1 + (k-1)\beta_1) + (k\lambda_0 + (k-1)\lambda_1)(\mu_D + \lambda_2 + (k-1)\lambda_1)$
$V(x) = (k\lambda_0(k-1)\lambda_1(\alpha_1 + \lambda_2 + (\mu_D(k-1)\lambda_1)(k-1)(\lambda_1 + \beta_1)$

If the faulty node replacement rate $\mu_D- > \infty$ then the simplified formula for MTTDL is

(13)

$$MTTDL = \frac{\alpha_1 + k\lambda_0 + (k-1)(\lambda_1 + \beta_1)}{k\lambda_0(k-1)(\lambda_1 + \beta_1)} \tag{14}$$

If the faulty node replacement rate $\mu_D- > 0$ then the simplified formula is

$$MTTDL = \frac{1}{k\lambda_0} + \frac{1}{(k-1)\lambda_1} \tag{15}$$

Meantime to data loss occurs on the unavailability of data and failure of node capacity. MTTDL is decreasing when the node failure rate in the network has been increased.

IV. Performance Evaluation for DRAFT Algorithm

This section presents the simulation setup, performance metrics, comparison of performance analysis with and without the DRAFT algorithm incorporated into the network.

A. Simulation Setup

The proposed DRAFT algorithm has been executed in CUPCARBON. The simulation setup consists of 150 sensor nodes with 27 clusters and is randomly deployed in a 200 X 200m square region. Assuming that all sensor nodes are to be homogeneous resources and characteristics.Each node data size is 100 to 2000 units and generates the data items periodically. Every node in the cluster maintains a DNN attribute table which holds MAC address, neighbors list, node id and the storage space of each node learned through continuous resource management messages broadcast by every 10s. Each round simulation time is set to be 600s. Fault-tolerant system parameters are listed in the following Table III.

TABLE III. FAULT-TOLERANT SYSTEM PARAMETERS

Fault-tolerant system parameters	
No. of Nodes	100-150 Nodes
Size of Data	100 to 2000 units
Sensing interval	[1-5]s
Broadcast interval	10s
Round simulation time	600s
Sensing field area	200 X 200m square
Node failure model	Random
MAC Protocol	802.15.4
Propagation loss model	log distance
Network Topology	Random

B. Performance Metrics

The impact of the DRAFT algorithm on an IoT network is analyzed through the following performance metrics.

1) Network Lifetime: In the simulation, the average Network lifetime of the IoT network with and without the DRAFT algorithm has been evaluated. This algorithm recovers the data when node failure occurs in the network. Considering 'p' is the probability of node failure that a node fails in one round. Assuming that the probability of a node failure for each round should vary from 0.1% to 0.5% as an increment. The total no. of deployed nodes in a cluster is 'Ni' and the communication range is 'X'. For without recovery scheme, in around at least a single node failure probability is P_f and is $1-(1-p)^(Nd)$ (Bernoulli's trails) whereas Nd is the network density and considering as a standard value.In the case of with recovery scheme, 'R' is the recovery candidate, and there are two requirements to recover the data, first, the recovery candidate must be alive and the second is all of its direct neighbors should be alive. The probability of R is $P_V = (1-p)^(Nd)$ whereas P_v is the probability of a recovery candidate. The

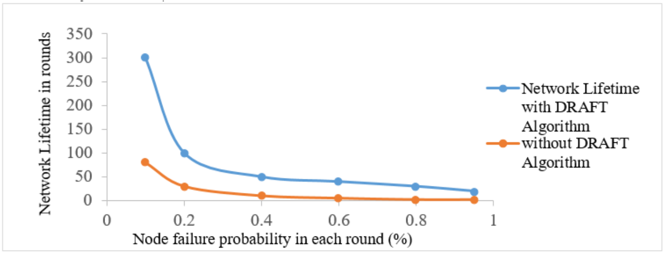

Fig. 4. Analysis of Network Lifetime with Respect to Node Failure Probability.

network lifetime of the proposed recovery scheme is more as compared to the without DRAFT algorithm as clearly shown in Fig. 4. When the probability of node failure is increased automatically the network lifetime is dropped for both with and without recovery scheme.

2) Throughput: Throughput is defined as the amount of data transmitted successfully from one node to another node in the network within a period. When the probability of the node failure decreases then the throughput of the network increases gradually as shown in Fig. 5. If there is a node failure occurs the data has been transmitted successfully because of the DRAFT algorithm. The throughput has improved double times with the recovery scheme as compared to the without recovery scheme.

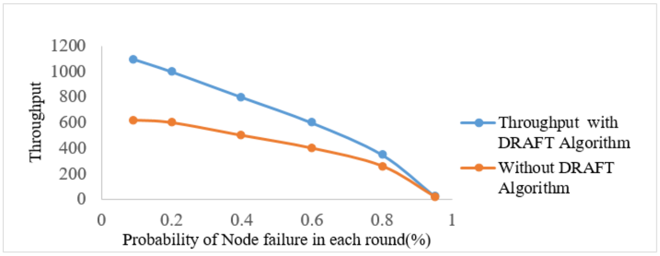

Fig. 5. Throughput with Respect to Probability Node Failure.

3) MTTDL: Meantime to data loss is the average time that causes data loss in the node. Data loss is occurred due to error situations in the networks. Backup and data recovery methods are helping to recover data or to avoid data loss in the IoT networks. If failure of any node in the cluster,

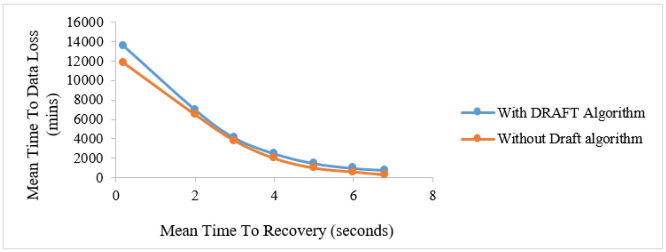

Fig. 6. Simulation Result of Mean Time to Data Loss.

MTTDL is decreased by increasing the recovery rate of the data. This will help to improve the network reliability and data availability in the network. The simulation results in Fig. 7 show the best recovery rate when the DRAFT algorithm has been incorporated into the network.

4) Reliability: Reliability is the capacity of the network to work during the presence of node failures concerning time. Here the time considering as the normalized time which means scaling the time within the range of 0 to 1. For an IoT network

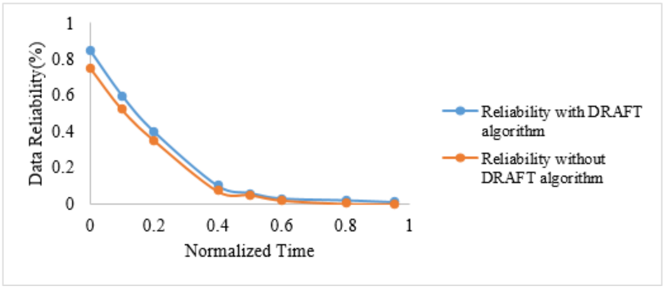

Fig. 7. Observations for Data Reliability in IoT Network.

at t = 0, the reliability is approximately high, with respect to time the network reliability is gradually decreasing which is shown in Fig. 6. The reliability mainly depends on the failure rate and repair rate of the node, and data recovery of the node.

V. Conclusion and Future Work

The proposed DRAFT algorithm is most suitable for IoT-based safety and critical applications. It is implemented with the concept of the RAID5 storage mechanism used for soft computing. Hence, this scheme can be easily incorporated with other IoT algorithms. By conducting a series of experiments the data reliability achieved as 85%. The throughput and network lifetime of the proposed algorithm are enhanced 5% as compared with the existing algorithms. The complete data recovery simulation is carried out which ensures reliable data transmission. In real scenario the data can be dropped due to noise, environmental factors, and data collisions. This paper presented analysis and simulation of simultaneous single IoT node failure in a cluster for the mentioned scenarios. This algorithm can be carry forward for the future multi-node data failures and data transmission reliability.

References

[1] Hong Chen, David Hailey, Ning Wang, and Ping Yu. A review of data quality assessment methods for public health information systems. *International journal of environmental research and public health*, 11(5):5170–5207, 2014.

[2] Xianke Sun, Gaoliang Wang, Liuyang Xu, and Honglei Yuan. Data replication techniques in the internet of things: a systematic literature review. *Library Hi Tech*, 2021.

[3] Samaresh Bera, Sudip Misra, and Athanasios V Vasilakos. Software-defined networking for internet of things: A survey. *IEEE Internet of Things Journal*, 4(6):1994–2008, 2017.

[4] Argyro Mavrogiorgou, Athanasios Kiourtis, Chrysostomos Symvoulidis, and Dimosthenis Kyriazis. Capturing the reliability of unknown devices in the iot world. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 62–69. IEEE, 2018.

[5] Mohamed Younis, Izzet F Senturk, Kemal Akkaya, Sookyoung Lee, and Fatih Senel. Topology management techniques for tolerating node failures in wireless sensor networks: A survey. *Computer Networks*, 58:254–283, 2014.

[6] Chunsheng Zhu, Lei Shu, Takahiro Hara, Lei Wang, Shojiro Nishio, and Laurence T Yang. A survey on communication and data management issues in mobile sensor networks. *Wireless Communications and Mobile Computing*, 14(1):19–36, 2014.

[7] Kun Wang, Yun Shao, Lei Xie, Jie Wu, and Song Guo. Adaptive and fault-tolerant data processing in healthcare iot based on fog computing. *IEEE transactions on network science and engineering*, 7(1):263–273, 2018.

[8] Yuchang Mo, Liudong Xing, Wenzhong Guo, Shaobin Cai, Zhao Zhang, and Jianhui Jiang. Reliability analysis of iot networks with community structures. *IEEE Transactions on Network Science and Engineering*, 7(1):304–315, 2018.

[9] Hui Yie Teh, Andreas W Kempa-Liehr, I Kevin, and Kai Wang. Sensor data quality: a systematic review. *Journal of Big Data*, 7(1):1–49, 2020.

[10] Ericsson. Ericsson 50 billion connections. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 434–462. IEEE, 2020.

[11] James Hong, Johnathan Raymond, and Joel Shackelford. Edisense: A replicated datastore for iot data. *Stanford University, Stanford*, 7(2):1–49, 2014.

[12] Pietro Gonizzi, Gianluigi Ferrari, Vincent Gay, and Jérémie Leguay. Data dissemination scheme for distributed storage for iot observation systems at large scale. *Information Fusion*, 22:16–25, 2015.

[13] Akshay Kumar, Nanjangud C Narendra, and Umesh Bellur. Uploading and replicating internet of things (iot) data on distributed cloud storage. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 670–677. IEEE, 2016.

[14] Juan A Colmenares, Reza Dorrigiv, and Daniel G Waddington. A single-node datastore for high-velocity multidimensional sensor data. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 445–452. IEEE, 2017.

[15] Waleed Bin Qaim and Öznur Özkasap. State-of-the-art data replication techniques in iot-based sensor systems. In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2018.

[16] Waleed Bin Qaim and Oznur Ozkasap. Draw: Data replication for enhanced data availability in iot-based sensor systems. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 770–775. IEEE, 2018.

[17] Chao Wang, Christopher Gill, and Chenyang Lu. Adaptive data replication in real-time reliable edge computing for internet of things. In *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 128–134. IEEE, 2020.

[18] Tzung-Shi Chen, Neng-Chung Wang, and Jia-Shiun Wu. An efficient adjustable grid-based data replication scheme for wireless sensor networks. *Ad Hoc Networks*, 36:203–213, 2016.

[19] Berihun Fekade, Taras Maksymyuk, Maryan Kyryk, and Minho Jo. Probabilistic recovery of incomplete sensed data in iot. *IEEE Internet of Things Journal*, 5(4):2282–2292, 2017.

[20] Yushi Shi, Xiaoqi Zhang, Qiaohong Hu, and Hongju Cheng. Data recovery algorithm based on generative adversarial networks in crowd sensing internet of things. *Personal and Ubiquitous Computing*, pages 1–14, 2020.

[21] Chinmaya Mahapatra, Zhengguo Sheng, Victor CM Leung, and Thanos Stouraitis. A reliable and energy efficient iot data transmission scheme for smart cities based on redundant residue based error correction coding. In *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking-Workshops (SECON Workshops)*, pages 1–6. IEEE, 2015.

[22] Teng Xu and Miodrag Potkonjak. Energy-efficient fault tolerance approach for internet of things applications. In *Proceedings of the 35th International Conference on Computer-Aided Design*, pages 1–8, 2016.

[23] Chen Wang, Hoang Tam Vo, and Peng Ni. An iot application for fault diagnosis and prediction. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 726–731. IEEE, 2015.

[24] Sen Zhou, Kwei-Jay Lin, and Chi-Sheng Shih. Device clustering for fault monitoring in internet of things systems. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 228–233. IEEE, 2015.

[25] Enver Derun Karabeyoğlu and Tufan Coşkun Karalar. Iot module improves smart environment reliability. In *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*, pages 1–5. IEEE, 2017.