DeCoRIC: Decentralized Connected Resilient IoT Clustering

Nitin Shivaraman[1], Saravanan Ramanathan[1], Shreejith Shanker[2], Arvind Easwaran[3], Sebastian Steinhorst[4]
[1]TUMCREATE Limited, Singapore, [2]Trinity College Dublin, Ireland, [3]Nanyang Technological University, Singapore, [4]Technical University of Munich, Germany
Email: [1]{nitin.shivaraman, saravanan.ramanathan}@tum-create.edu.sg, [2]shankers@tcd.ie, [3]arvinde@ntu.edu.sg
[4]sebastian.steinhorst@tum.de

***Abstract*—Maintaining peer-to-peer connectivity with low energy overhead is a key requirement for several emerging Internet of Things (IoT) applications. It is also desirable to develop such connectivity solutions for non-static network topologies, so that resilience to device failures can be fully realized. Decentralized clustering has emerged as a promising technique to address this critical challenge. Clustering of nodes around cluster heads (CHs) provides an energy-efficient two-tier framework for peer-to-peer communication. At the same time, decentralization ensures that the framework can quickly adapt to a dynamically changing network topology. Although some decentralized clustering solutions have been proposed in the literature, they either lack guarantees on connectivity or incur significant energy overhead to maintain the clusters. In this paper, we present Decentralized Connected Resilient IoT Clustering (DeCoRIC), an energy-efficient clustering scheme that is self-organizing and resilient to network changes while guaranteeing connectivity. Using experiments implemented on the Contiki simulator, we show that our clustering scheme adapts itself to node faults in a time-bound manner. Our experiments show that DeCoRIC achieves 100% connectivity among all nodes while improving the power efficiency of nodes in the system compared to the state-of-the-art techniques BEEM and LEACH by up to 110% and 70%, respectively. The improved power efficiency also translates to longer lifetime before first node death with a best-case of 109% longer than BEEM and 42% longer than LEACH.**

***Index Terms*—IoT, Clustering, Resiliency, Decentralization.**

I. Introduction

The Internet of Things (IoT) enables millions of nodes (devices) to exchange information to form intelligent connected systems. However, IoT networks exhibit some unique traits compared to 'traditional' distributed systems. The nodes of an IoT network are primarily low-cost and resource-constrained, which may join or leave the network in an ad-hoc manner and communicate over a wireless interface. Based on the information exchanged among neighboring nodes, each node may take independent decisions enabling decentralized operation.

Information exchange over a wireless medium necessitates the design of energy-efficient communication strategies for these nodes to extend their lifetime. Furthermore, it is a challenge to maintain end-to-end connectivity among the nodes of the network in the presence of changes in the communication links due to the ad-hoc nature of the network.

This work was financially supported in part by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme. With the support of the Technische Universität München - Institute for Advanced Study, funded by the German Excellence Initiative and the European Union Seventh Framework Programme under grant agreement n° 291763

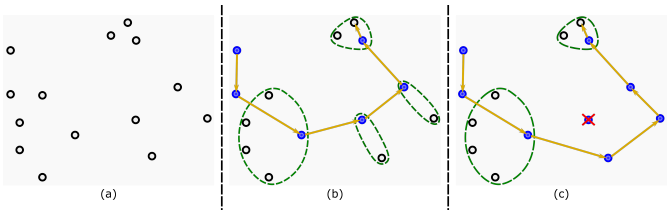

Fig. 1: Clustering operation: starting with any topology (a), the nodes align themselves into clusters with an elected CH (b). With DeCoRIC, the clustering dynamically adapts to changes in topology to ensure connectivity among all nodes (c).

Clustering has been shown as the most effective technique to improve energy efficiency and scalability in networked systems [1]. Nodes, as illustrated in Figure 1 (a), are grouped into *clusters* based on common node properties such as residual energy, location or degree (number of communication links of the node). Cluster sizes can be equal or unequal depending on the chosen property. Nodes marked in blue are elected representative nodes called cluster heads (CHs), each of which acts as the data aggregator and nodal point for multi-hop communication, as shown in Figure 1 (b), allowing regular (non-CH) nodes to operate in low-power mode more often to conserve energy.

Most clustering techniques aim at improving the lifetime of the network, but often result in disassociated clusters and nodes operating independently without being able to establish any communication with neighbors, leading to loss of data and connectivity. Also, static topologies lack the flexibility to deal with the ad-hoc nature of an IoT network as well as with node failures during operation. We believe that the following properties are essential in any IoT clustering technique to complement the existing capabilities:

1) **Connectivity** – the clustering technique must ensure that the nodes are clustered such that there is a path between any two nodes in the network whenever possible; this property ensures reliable routing of information between any two nodes in the network.
2) **Decentralization** – each node must make independent decisions without a central entity; this ensures there is no single-point of failure.
3) **Resilience** – the network must adapt to node faults or network changes at run-time by detecting and reorganizing in a time-bound manner to ensure *connectivity*.

In this paper, we propose Decentralized Connected Resilient IoT Clustering (DeCoRIC), a clustering scheme that can group

nodes into connected clusters and adapt to network changes at runtime without relying on a central node or prior information (topology, position, etc.). Each node takes decisions independently and collectively manages the clustering process to achieve the above goals. Based on the information gathered from their neighbors, nodes make decisions and react to topology changes by altering their state, as seen in Figure 1 (c), to ensure connectivity, while minimizing energy overheads.

LEACH [1] and BEEM [2] are chosen as the representative schemes for comparison. LEACH is the de-facto benchmark of clustering algorithms, while BEEM is a recent extension of another benchmark, Hybrid Energy-Efficient Distributed clustering (HEED) [3], aimed at higher connectivity. We evaluate DeCoRIC using multiple random network topologies to show that the above properties are achieved, while also enabling up to 70% and 110% improvement in power efficiency over BEEM and LEACH, respectively.

This paper presents the following contributions: (i) We propose DeCoRIC, our scheme for Decentralized Connected Resilient IoT Clustering (Section III). (ii) We have ported the state-of-the-art techniques LEACH and BEEM into the Contiki simulator to emulate a realistic communication environment (Section IV) for comparison with DeCoRIC and made the implementations open-source. (iii) We show through results (Section IV) that DeCoRIC converges to a resilient fully connected network with bounded latency.

II. Related Work

Radio communication is a key component that largely influences the energy consumption in IoT nodes. Clustering techniques aim at reducing this energy consumption by partitioning the network into clusters. Each cluster has an active cluster head (CH) as a representative node elected by either by a central entity or all the nodes in the cluster. Clustering enables the regular (non-CH) nodes to reduce the frequency of transmission and operate in low-power mode, reducing the overall power consumed by the system. Cluster head oversees the multi-hop communication and performs data fusion resulting in minimal communication for the non-CH nodes. Clustering techniques can be classified into centralized and decentralized methodologies, based on whether the clustering decision and CH election is performed by a central entity or independently by nodes of the network.

Centralized Clustering: Centralized techniques rely on a central entity that has global knowledge of the network, and manages the CH election and clustering process. The clustering operation can be based on the degree of a node in the network [7], residual energy of nodes [8] or other parameters. The degree of a node is defined as the number of neighbors within the radio range. The clustering problem was formulated as a linear programming problem in [9], representing a trade-off between energy consumption and the quality of communication. A centralized version of a popular decentralized algorithm, LEACH (described below), was developed in [10], where control decisions are managed by a central entity, making more efficient CH selection than LEACH. Further improvement was made in [4], where nodes are organized into a chain based on proximity to evenly distribute the transmission energy. While there is no deterministic polynomial algorithm that can partition a network topology into clusters [11], meta-heuristic algorithms like particle swarm optimization and artificial bee colony have been successfully applied in the clustering of wireless networks [12], [13]. The requirement of a central entity (often the base station) in centralized systems results in higher clustering latency and scalability issues, since every decision has to be relayed from the central entity. The centralized approach also results in a single point of failure at the central node, inhibiting effective ways to enable resilience and connectivity. Distributing the clustering operation among nodes aims to mitigate some of these issues, albeit centralized techniques generally offer superior energy efficiency over decentralized strategies.

Decentralized Clustering: HEED was one of the earliest decentralized techniques and uses a combination of node degree and residual energy as the metric for clustering [3]. BEEM [2] is the most recent extension of HEED that includes node degree in the CH election conditions to improve connectivity by letting nodes in denser areas expend higher energy.

Low-Energy Adaptive Clustering Hierarchy (LEACH) [1] is the most popular decentralized technique, which used probabilistic election of a CH and its rotation within a cluster to ensure uniform energy distribution. Enhancements to the LEACH protocol that enable power optimization through two-level adaptive clustering [14], and multi-level hierarchical clustering [15] have also been proposed. More recent enhancements to the protocol added residual energy [16] and multi-hop communication [17] in the CH election process to achieve minor improvements in energy and throughput, respectively. LEACH and HEED are used as benchmarks in clustering by the community [18]. As BEEM extends HEED ensuring connectivity, LEACH and BEEM are chosen as representative techniques in our paper for comparison with DeCoRIC.

Other notable works include overlapping clusters [19], [20] where nodes belong to multiple clusters simultaneously to ensure connectivity among the clusters. Unequal clusters [6] were used to reduce the impact of high activity for nodes close to the base station. The work in [5] form multi-level clustering using overhearing characteristics of the wireless medium to form clusters adaptively. Event-based clustering schemes such as Bee-Sensor-C [21] perform a local clustering around an event (such as sensor value change) but suffer from poor energy efficiency without any clustering for non-event nodes. Methods such as [15] use multi-hop within the cluster, causing high node activity and energy consumption, while also presenting challenges in reliable message delivery and clustering convergence when the network scales. Techniques that employ overlapping for connectivity [19], [20] are susceptible to hidden node collision faults, affecting reliable message exchange. However, most of the existing decentralized schemes use a fixed network topology and cannot cater to dynamic ad-hoc networks of the IoT. Further, most techniques do not consider connectivity across all nodes and often result in isolated clusters, albeit the nodes are within the radio range of each other. A summary of some works and their properties is shown in Table I.

Additionally, there is a body on literature which look into clustering in a graph theoretic perspective [22], [23],

TABLE I: Comparison of notable works in Literature.

Property/ protocol	LEACH [1]	PEGASIS [4]	HEED [3]	PEACH [5]	EEUC [6]	BEEM [2]	DeCoRIC
Location/ topology data	No	Yes	No	Yes	Yes	Yes	No
Centralized/ Decentralized	Decentralized	Centralized	Decentralized	Decentralized	Decentralized	Decentralized	Decentralized
Complexity	Low	High O(N^2)	Low	High O(N^2)	Low	Low	Low
Clustering mechanism	Residual energy	Location	Residual energy	Proximity (overhearing)	Residual energy and Base station proximity	Residual energy and Degree	Degree
Communication channel	TDMA	CDMA	TDMA	TDMA	TDMA	TDMA	CSMA
Connected clusters	No	No	Yes	No	Yes	Yes	Yes
Resilience to failures	No	No	No	No	No	Yes	Yes

[24]. The network is mapped as a unit disk graph to find the minimum connected dominating set (MCDS) for various network topologies. However, the literature in this direction is unrelated to the presented algorithm in this paper as they do not consider any radio model in the network, leading to a theoretical solution that may not be practically viable.

To the best of our knowledge and as reported in the literature [18], there is no existing clustering method that combines the three properties of decentralization, connectivity and resilience.

III. DeCoRIC strategy

In this section, we will introduce the network assumptions as well as the detailed clustering phases of DeCoRIC.

A. IoT Network Assumptions

We make the following assumptions about the IoT network:

1) The network uses wireless communication among the nodes based on the IEEE 802.15.4 standard [25].
2) Carrier-sense multiple access with collision avoidance (CSMA/CA) is employed at the MAC layer to mitigate congestion for broadcast messages.
3) Each node operates independently without any central entity or apriori information about the topology (i.e., fully decentralized network).
4) Nodes are equipped with a processor, clock, memory and a unique identification (ID) used for book-keeping.
5) The network is ad-hoc where nodes can join or leave the network at runtime (e.g. node failures).
6) All nodes transmit and receive on the same channel with the same signal strength asynchronously.
7) The network uses a hard fault failure model, i.e., a faulty node ceases to transmit information on the network.

B. Clustering Strategy

DeCoRIC operates independently at each node in four phases defined by a Finite State Machine (FSM) as shown in Figure 2 with each phase lasting for a pre-configured period (called *round*, discussed below). The nodes power up in the *Discovery* phase, where each node waits to receive messages from its neighboring nodes and evaluate its environment. The transition occurs at the end of the period to the *Election* phase, where the node that has the highest degree declares itself as a CH followed by neighboring nodes associating themselves to nearby CHs forming clusters. Progression to the *Correction* phase after Election phase initiates evaluation of the connectivity property to identify isolated clusters/nodes. Non-CH nodes within a cluster, which can enable connectivity between two CH nodes that are out of range, break out to form *Bridge-CH* nodes. The system transitions into the *Stable* phase at the end of the Correction phase, where the nodes periodically check the status of their neighbors by exchanging *health* information. If changes are detected (i.e., failures or new nodes in the system), the nodes go back to the Election phase and follow the path to re-establish a stable operation. Once in the Stable phase, a Bridge-CH could upgrade itself to a full CH, if newly joining nodes affiliate themselves with the Bridge-CH, forming new clusters.

To enable this operation, DeCoRIC uses broadcast messages as payload, shown in Figure 3, that is encapsulated in a regular IEEE 802.15.4 frame. The *Node ID* field marks the ID of the transmitter and is always present in all messages. Only the Node ID field is valid in the Discovery phase as there is no information about the neighboring nodes. In the subsequent

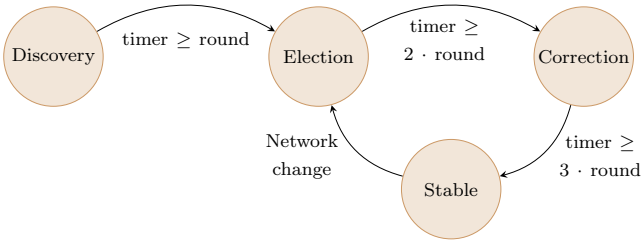

Fig. 2: DeCoRIC phases and transition conditions.

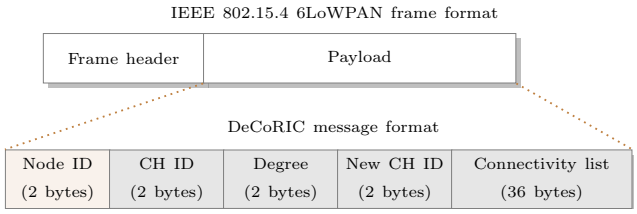

Fig. 3: DeCoRIC message format.

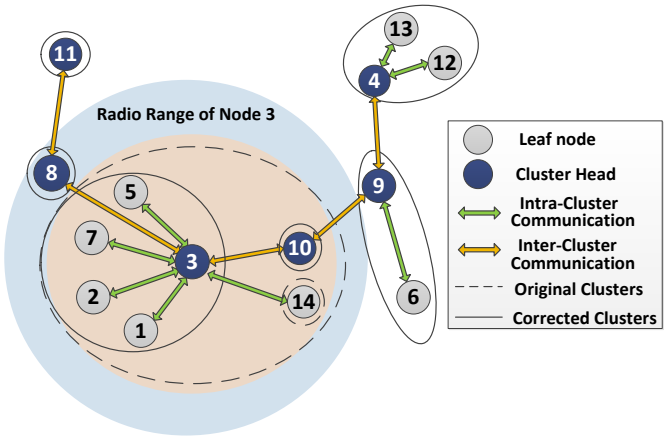

Fig. 4: DeCoRIC on an example network (not drawn to scale).

phases, the *CH ID* and degree become known to each node, which feeds into the *neighbor list* (i.e., a list of all neighbors a node has received direct messages from). While the neighbor list contains all neighbors that are in range of the node, the actual connected nodes are maintained using a second list called the *connectivity* list. The connectivity list gets updated periodically, reflecting the activity of connected nodes. This two-level list structure allows DeCoRIC to eliminate false positives on the propagation of the activity of the nodes during the Stable phase (discussed in Section III-B4). The *New CH ID* field is used only when a new node determines that it has to be the CH as it attained a higher degree than its current CH. The operation details of each node as it transitions through the DeCoRIC phases are described below.

1) Discovery phase: The neighbor discovery phase enables each node to discover its neighboring nodes that it can communicate with and, hence, its own degree. The steps involved in the Discovery phase are listed in Algorithm 1. In this phase, each node sends a DeCoRIC *ping* message with only the node ID and CH-ID fields filled with its own identifier (others left as zeros). All nodes keep their radio active during this phase to receive messages from their neighbors.

The RSSI threshold is a configurable parameter to ensure that communication links among the nodes can offer sufficient signal to noise ratio (SNR) [26] for reliable communication. A receiving node updates its degree with each received message and updates the neighbor list. Nodes that meet a received signal strength indicator (RSSI) threshold are marked as a *potential* neighbors that could belong to the same cluster. Nodes that fail to meet the threshold do not belong to the same cluster and are marked as external neighbors.

For the example network shown in Figure 4, node 3 has a communication range of the blue shaded area while the RSSI threshold limits the cluster range to the orange shaded area. Node 3 receives messages from 1, 2, 5, 7, 8, 10 and 14 leading to a degree of 7. Node 8 is marked as an external neighbor while nodes 1, 2, 5, 7, 10 and 14 are marked as potential neighbors based on the configured RSSI threshold. Node 3 includes these 7 neighbors in its neighbor list at the end of this phase. Similarly, node 9 marks 6 as a potential neighbor,

Algorithm 1 Neighbor Discovery phase

```
LIST: Neighbor, Conn = FALSE
Degree = 0, CH.ID = node.ID
broadcast(Msg)
if rcv() then
    Msg = rcv().data
    Neighbor[Msg.ID], Conn[Msg.ID] = TRUE
    Degree = Degree + 1
    if rcv().RSSI < RSSI_threshold then
        Neighbor[Msg.ID].ext = TRUE
    end if
end if
```

while 4, 10, 12, 13, 14 become external neighbors to 9. Nodes 12 and 13 are marked potential neighbors by node 4, with node 9 as an external neighbor. Node 11 is out of the RSSI threshold range of node 8 but within the radio range. Thus, node 8 marks 11 as an external neighbor.

Since the nodes in the network communicate asynchronously, multiple nodes will attempt to transmit during any given time, resulting in collisions. Although the use of CSMA-CA avoids collisions, it is important that a node is able to complete the transmission without failures or indefinite wait times due to back-off. To ensure that each node can transmit at least one DeCoRIC message in the Discovery phase, the transmission time window is computed based on the IEEE 802.15.4 standard [27] including the worst-case back-off. This value is aggregated over the maximum number of nodes supported by the network to form a time window referred to as *round* in DeCoRIC. The nodes stay in the Discovery phase for one round, which ensures that all nodes have successfully transmitted at least one DeCoRIC *ping* message. It is important to note that all nodes are asynchronous. The time duration of one round is calculated as:

$$round = N \cdot \Big(\sum_{i=0}^{maxR} \tau_{\mathrm{bo,i}} + \tau_{\mathrm{fr}} + \tau_{\mathrm{ifs}} \Big) \tag{1}$$

where, $\tau_{\mathrm{bo,i}} = \big(2^{\,maxBE(i)} - 1\big) \cdot \tau_{\mathrm{symb}} + \tau_{\mathrm{cca}}$,

N is the maximum number of nodes in the network, $maxR$ is the maximum number of retries, $\tau_{\mathrm{bo,i}}$ is the worst-case back-off delay at the i^{th} retransmission, τ_{fr} is the frame transmission time, τ_{ifs} is the minimum inter-frame period, $maxBE$ is the maximum back-off parameter at the i^{th} retransmission, τ_{symb} is the back-off symbol period and τ_{cca} is the clear channel assessment time. The parameters τ_{ifs}, τ_{symb} and τ_{cca} are derived from the network standard (IEEE 802.15.4), while $maxBE$, N and τ_{fr} are configured with the same value at each node (as network parameters).

2) Election phase: In this phase, nodes transmit a DeCoRIC message with an up-to-date degree field obtained from the previous phase. Each receiving node independently compares its own degree to the received degree to keep track of the node with the highest degree (potential CH). Once each node has received at least one transmission from each neighbor (ensured by round configuration), it sets the node with the highest degree as its CH. If multiple nodes have the same highest degree, the node with the lower node ID is chosen as

Algorithm 2 Cluster Election phase

```
1: broadcast(Msg)
2: if rcv() then
3:     Msg = rcv().data
4:     if Msg.Degree > Degree then  CH.ID = Msg.ID
5:     else if (Msg.Degree == Degree) & (node.ID > Msg.ID) then
6:         CH.ID = Msg.ID
7:     end if
8: end if
```

Algorithm 3 Cluster Correction phase

```
1: broadcast(Msg)
2: if rcv() then
3:     Msg = rcv().data
4:     if (Msg.CHID == Msg.ID) AND (Msg.CHID != CH.ID) then
5:         if Msg.Conn[CH.ID] == 0 then
6:             CH.ID = node.ID  /* Detach from cluster */
7:         end if
8:     end if
9: end if
```

CH. The configuration can be altered to choose a higher ID or support priority for specific node IDs. The operation of this phase is described in Algorithm 2. Similar to the discovery phase, all nodes keep the radio active during this phase. A message from a new node will be updated into the neighbor list and the connectivity list, while messages from existing nodes reinforce their active state in the connectivity list.

In the case of the example system in Figure 4, node 3 becomes a CH with nodes 1, 2, 5, 7, 10 and 14 as its members at the end of this phase. Node 9 becomes a CH with node 6 as a member, node 4 forms the CH with nodes 12 and 13 as members, while nodes 8 and 11 become independent CHs.

3) Correction phase: Once the clusters are established, there exists the possibility of isolated clusters, i.e., the Cluster Heads are not within each others' radio range, but some common nodes of either cluster can connect the two clusters. To prevent isolated clusters, each non-CH node verifies the **connectivity** property based on the connectivity list and root-ID fields of the received messages. If any non-CH node satisfies the connectivity property, it breaks out from the affiliated cluster to form a Bridge-CH. This correction process is described in Algorithm 3. If multiple nodes can enable connectivity between the same set of CHs, the rule for CH election is followed (see Section III-B2). Redundant bridge nodes continue to operate as non-CH nodes, reducing interference to the existing bridge nodes during inter-cluster communication and, thereby, minimizing their power consumption. Once all the nodes verify the connectivity property, the clusters and CHs established are finalized and the network moves to a stable execution phase.

Referring back to the example in Figure 4, nodes 10 and 14 identify that they can enable direct connectivity between CH node 9 and their current CH node 3 based on information from the connectivity list. As node 10 has the same degree as node 14, node 10 breaks out as the Bridge-CH because of its lower ID, while node 14 continues as cluster member.

The correction phase ensures that non-CH nodes strictly remain in low-power mode without involvement in inter-cluster communication while connectivity among nodes is ensured by CH nodes. Further, this phase minimizes energy overhead and latency in inter-cluster communication by enabling single-hop connection among CHs, while lowering congestion and error propagation at the bridge-CH interfaces (hidden node collision problem) [28]. A node which breaks out from a cluster will not attempt to reintegrate into a cluster and remains an independent cluster head unless a network change invalidates the correction. This ensures that the algorithm converges to an operative network condition at each node in a time-bound manner without frequent re-clustering.

4) Stable phase: In the Stable phase, the CH nodes broadcast a fully populated DeCoRIC frame as *health* message every round. Health message informs the other nodes that the sender node has not exhausted its energy. They also serve as a re-clustering trigger if there is a change in the network topology. All nodes activate radio duty cycling (RDC) [29] which keeps the receiver active only for a fraction of time in a periodic manner (determined by parameter RDC_{rate}) to reduce the power consumed by the radio. Thus, a successful transmission may not guarantee reception at each node. To address this, DeCoRIC defines a configurable period called *cycle* as the minimum set of rounds that will ensure that a non-CH node receives at least one transmission from its CH. Cycle duration is computed as:

$$cycle = h \cdot \text{LCM}(txn_{\text{freq}}/h, RDC_{\text{rate}}/h) \quad (2)$$
$$\text{where, } h = \text{GCD}(txn_{\text{freq}}, RDC_{\text{rate}}),$$

txn_{freq} is the round duration, RDC_{rate} is the RDC frequency in number of ON periods per second, GCD() and LCM() are the greatest common divisor and least common multiple functions, respectively.

Non-CH nodes aggregate the received CH health messages over a cycle and acknowledge with a health message at the end of the cycle. The per-cycle message from non-CH nodes not only reduces network traffic, but also conserves energy at these nodes. The connectivity list aids in failure detection and gets updated upon receiving health messages.

Failure Detection: Following the Discovery, Election and the Correction phases, all the nodes switch to the RDC mechanism and have different transmission intervals depending on their CH/non-CH status. Furthermore, as the nodes are not synchronized, the sleep windows at each node might be different. Thus, transmissions from a node may be missed by its neighbors, leading to false positives about a node's state.

To overcome this, DeCoRIC employs a modified gossiping scheme, derived from [30], to spread information about the node health. Each node maintains a *fail* counter (counting rounds) T_{fail} for every node in its neighbor list and it is incremented at every round. Any node which receives a *direct* message from a neighbor node resets the corresponding fail counter to zero and gossips about the health of the neighbor node by including its ID in its connectivity list. Meanwhile, if a node receives a *gossip* message about a neighbor node (i.e., from the connectivity list of a received message), the fail counter corresponding to that node is reduced by half. When

Algorithm 4 Stable phase

```
1: if rcv() then
2:     Msg = rcv().data
3:     if Msg.Degree >CH.Degree then
4:         CH.ID = Msg.ID
5:         Phase = ELECTION
6:     end if
7:     for each item i in Conn do
8:         if (Msg.Conn[i] & Conn[i]) then
9:             fail[i] = 0
10:        else if (Msg.Conn[i] & !Conn[i]) then
11:            fail[i] = 0.5 · fail[i]
12:        end if
13:    end for
14:    Neighbor[Msg.ID], Conn[Msg.ID] = TRUE
15: end if
16: if round then
17:    for each item i in Conn do
18:        if fail[i] ≥ T_fail then Conn[i] = FALSE
19:        end if
20:        if fail[i] ≥ 2 · T_fail then Neighbor[i] = FALSE
21:        else fail[i] = fail[i] + 1
22:        end if
23:    end for
24:    if node.ID == CH.ID then broadcast(Msg)
25:    else if cycle then broadcast(Msg)
26:    end if
27: end if
```

the fail counter corresponding to a neighbor reaches T_{fail} at a node, the neighbor ID is marked faulty and removed from the connectivity list and, thus, its health message.

Assuming the hard fault model, once the fail counter reaches $2 \cdot T_{\text{fail}}$, the corresponding node's ID is marked as failed and removed from its neighbor list. The stable phase operation and the failure detection process is shown in Algorithm 4. The gossiping scheme helps to accommodate false triggers caused by missed packets or transient network conditions, at the expense of increased detection time for node failures.

Network Adaptation: A failure of a node or addition of new nodes creates a change in the clustering of the network. The change ranges from a few clusters (new node addition or non-CH failure) to the entire network (CH failures) depending on the connectivity of the failed node to other clusters. Only nodes whose degree change as a result of node failure switch to the Election phase; unaffected nodes continue to operate in the Stable phase. Finally, when a new node tries to integrate into an existing cluster (i.e., observing health messages), it joins into an existing cluster and could replace the current CH based on its degree that becomes apparent over the next cycle using the *New CH ID* field.

In our example network, when node 4 was deleted, it was observed that both nodes 12 and 13 go into the Election phase after the detection of the failure. After the Election phase, node 12 declares itself as the new CH while node 13 operates as non-CH in the new cluster.

IV. Analysis & Evaluation

In this section, we present the evaluation of DeCoRIC using the Cooja Simulator from Contiki [29]. We implemented LEACH and BEEM protocols on Contiki for comparison with DeCoRIC. We measure the average power consumption per node and the time for the first node death to compare the power efficiency of the protocols. The protocol with the least power consumption and the longest time to death for the first node would have the most power-efficient operation, assuming they offer similar connectivity. We also show the progression of nodes exhausting their energy over time to quantify the power distribution of the protocols among the nodes of the network. This experiment also gives a measure of time during which the network stays intact and connected. To further illustrate the connectivity, we reduce the range of nodes in the simulation to show the time taken and power expended by the protocols to achieve 100% connectivity. Additionally, our test scenarios analyze and evaluate the resilience of DeCoRIC by triggering faults in the network and quantifying the worst-case delay before the network stabilization post re-clustering.

Cooja Simulator: Contiki's Cooja Simulator allows development in native C language, which can then be directly deployed on a compatible hardware platform [29]. The software elements are cross-compiled to a target hardware, similar to an emulation flow. This enables the evaluation to consider actual hardware constraints such as memory limitations (to fit the algorithm), network errors such as packet-loss and interference, and actual bit-level transmission at the cost of slower execution time. We use the Skymote [31] as the target hardware platform and employ the powertrace tool in Contiki to measure the power consumption of the devices for all our experiments. Skymote uses a Texas Instruments CC2420 transceiver that complies with the 2.4 GHz IEEE 802.15.4 6LoWPAN standard with a bit rate of 250 kbps and a processor platform that supports sustained low-power mode.

LEACH and BEEM Implementation: The original LEACH and BEEM implementations were done in MATLAB which abstracts away the low-level communication details (hardware radio model). Also, the MATLAB implementations were inherently centralized since the simulation system has an overall view of the state of each node. Hence, we implemented LEACH and BEEM on Contiki based on the original protocol in MATLAB and the description in the papers [1], [2] [1]. At the lowest level, we use the Contiki radio model as a common platform for emulating LEACH, BEEM and DeCoRIC using the Cooja Simulator; the higher layers are the C implementations of the respective protocols. All the protocol implementations in C are available as open-source for research use [2].

The radio of the non-CH nodes on both LEACH and BEEM implementations are turned off once the clustering is complete, except when they have to transmit messages to the CH. Meanwhile, the radio of the CH is always kept on to receive messages from non-CH nodes of the cluster as described by the protocols. We translate this TDMA behavior into Contiki using the RDC mechanism.

Although both protocols differ in their CH election process, they have a cyclic re-clustering mechanism after a period defined as an *epoch* [1], [2]. An optimal value of the epoch is paramount for energy efficiency on both protocols. We

[1] For the first time in a decentralized system with hardware emulation.

[2] The source code is available at https://bitbucket.org/nitinshivaraman/clustering_contiki.

TABLE II: Parameters used in our experimental setup for evaluating DeCoRIC against LEACH and BEEM.

Parameter	Values used
Area (m^2)	100x100
Number of nodes (N)	{50, 100, 200}
Transmission Range (m)	50
CDMA MAC Protocol	CSMA-CA (CXMAC), TDMA
Radio Frequency (GHz)	2.4
Topologies	{Random}
$maxBE$	3
Packet rate (packets/node/round)	1 round
RDC rate (activations/s)	32

conducted experiments to measure the power consumption by varying different epoch values. It was found that higher epochs yield a lower power consumption with CH nodes expending higher power, while shorter epochs lead to constant re-clustering and higher power expenditure from all nodes.

Experimental setup: All the experiments were performed on the Cooja simulator with different parameters. DeCoRIC, LEACH and BEEM are run for a group of 50, 100 and 200 nodes arranged in 100 random topologies in an area of 100 x 100 m^2. The topologies are common for all the three protocols, with nodes placed at random locations within a given area using the random placement feature of the Cooja simulator. The packet transmission rate is 1 packet/round with a transmission range of 50 m for each node. Round in DeCoRIC, is set based on Equation 1 as 0.8, 1.1 and 2.2 seconds, respectively, for 50, 100 and 200 nodes. The lower bound of 0.8 seconds is a restriction imposed by the simulator, below which transmission overlap was observed due to incomplete initialization, resulting in unintended collisions and data loss. One cycle is configured as 6 rounds, $T_{\text{fail}}^{\text{CH}}$ and $T_{\text{fail}}^{\text{nCH}}$ for CH and non-CH nodes are computed as 6 and 36 rounds, respectively, using Equations 1 and 2. To ensure that all three protocols achieve comparable stable state duration before the cyclic re-clustering process, the epoch was chosen to be 10 rounds. The simulation parameters used for our test setup are shown in Table II.

A. Power Consumption

Initially, we evaluate the change in average power consumption of the network and the resulting number of CH nodes by varying the RSSI value in DeCoRIC. Figure 5 shows the results of the experiment across different RSSI reception thresholds of -45 dBm, -65 dBm and -85 dBm represented by the x-axis for 100 nodes. The y-axis on the left represents the number of CHs formed while the average power consumption per node in milliWatts (mW) is shown on the y-axis to the right. The results show that in the case of -45 dBm nodes (lower effective radio range), less than 30% of nodes act as CH nodes on average with a worst-case of 43% CH nodes. This is a result of many nodes being marked as external neighbors in this case due to their positions in the topology.

As the RSSI threshold increases, DeCoRIC marks more nodes as potential neighbors, with a mean and worst case of 13 and 25 CHs at -85 dBm; a mean and worst case of 17 and 33 CHs at -65 dBm. In comparison, most clustering schemes (including LEACH and BEEM) predefine the number of CHs to be between 5–25% (see [8], [9], [10], [11]) of the total number of nodes with no consideration for the network

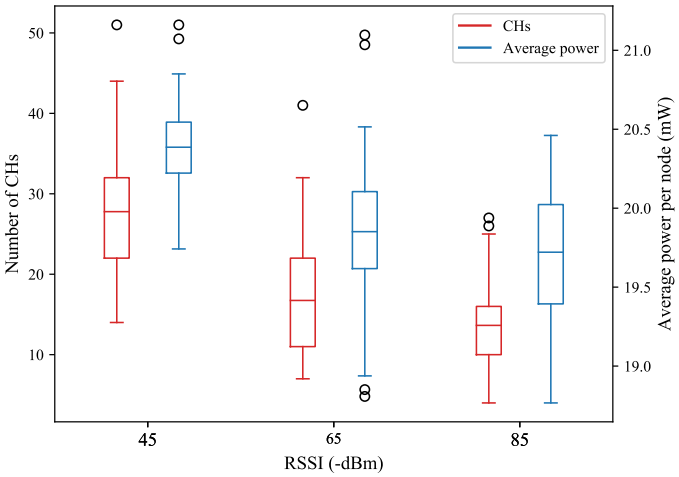

Fig. 5: Number of CH nodes and average power consumed by the nodes running DeCoRIC over different RSSI thresholds to form clusters.

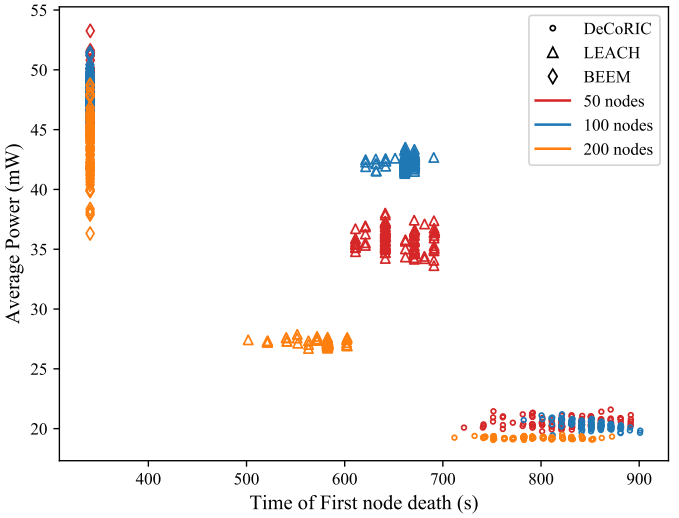

Fig. 6: Average Power Consumption vs Time of first node death in the network for DeCoRIC compared against LEACH and BEEM.

structure, often resulting in disconnected clusters. The outliers in the number of CHs for DeCoRIC can be attributed to the randomness of the topologies since nodes that are farther than the radio range of the RSSI threshold become members of different clusters.

Changing the RSSI results in a change of the cluster size, with higher RSSI leading to bigger clusters and lower RSSI leading to smaller ones. Larger clusters expend higher energy on CHs while reducing the overall network power consumption; smaller clusters result in higher network power consumption with many CH nodes as seen in Figure 5. The change in power is more pronounced as the number of nodes scales. Further, the outliers in the average power can be attributed to the fact that DeCoRIC employs bridge-CHs to facilitate connectivity in the network, which increases the average power consumption.

The RSSI threshold of -65 dBm is chosen for the rest of the

experiments for DeCoRIC. To demonstrate the power saving in DeCoRIC, first, we compare average power per node in milliWatts (mW) along with the time of their first node death (exhaust nodes' power completely) over a simulated duration of 1000 seconds for 50, 100 and 200 nodes in the network across LEACH, BEEM and DeCoRIC. Second, we record the time at which the nodes' death. To identify residual energy in the Contiki framework, the power model in [32] was integrated with *powertrace* with an initial battery capacity of 6mW (milli Watt Hour) to observe the energy drain of the nodes.

The results of the first experiment are shown in Figure 6, where the x-axis and y-axis represent the time for the first node death in seconds and the average power consumption per node in mW respectively. The number of nodes in the network is represented by different colors, while the protocols are represented by different shapes. From the results, it is seen that our proposed method has the least power consumption per node, thereby prolonging the time for the first node death. It offers a best-case of 70% and 110% improved power efficiency over LEACH and BEEM for 50 nodes. Similarly, the best-case improvement for the time of first node death is 42% and 109% over LEACH and BEEM for 200 nodes, respectively.

The energy savings in LEACH can be attributed to the proactive load distribution strategy with periodic re-clustering. Hence, there is a longer time before the first node dies as the power distribution is balanced. BEEM, on the other hand, adopts a strategy where CH nodes remain unchanged during re-clustering to retain connectivity, while non-CH nodes are retained in low-power mode during the periodic re-clustering. Due to this strategy, the CH nodes exhaust their power rapidly due to prolonged radio on-time. The periodic clustering in LEACH and BEEM results in higher power consumption for all the nodes. This is reflected in the plot where the total average power decreases as the number of nodes scales while the death of the first node happens faster.

By contrast, DeCoRIC uses a reactive strategy, reducing the activity in the Stable phase and re-clustering only for node failures. As there is no TDMA, all the nodes experience similar radio activity subject to the density of nodes in the network. Similar to LEACH and BEEM, there is a decrease in power consumption and faster depletion of node power as the network scales. Both BEEM and DeCoRIC retain the CH in order to ensure connectivity. However, DeCoRIC balances the radio activity efficiently with RDC, re-clustering only after node failures are detected and strives to achieve maximum connectivity. As seen from Figure 6, the power consumption and the active time of nodes are inversely related.

In order to maintain connectivity over a longer period, the power dissipation has to be managed efficiently among all the nodes. To quantify the rate of power consumption and the time of connectivity, we show the time at which the nodes exhaust their energies in a simulation of 1000 seconds. The time at which nodes die progressively is shown in Figure 7. The y-axis of the plot represents the number of nodes in the network at the start of the simulation while the x-axis represents the time in seconds.

Energy exhaustion rate of nodes is higher in LEACH than BEEM and DeCoRIC, as most nodes would have expended similar energies. BEEM has certain nodes in a denser area

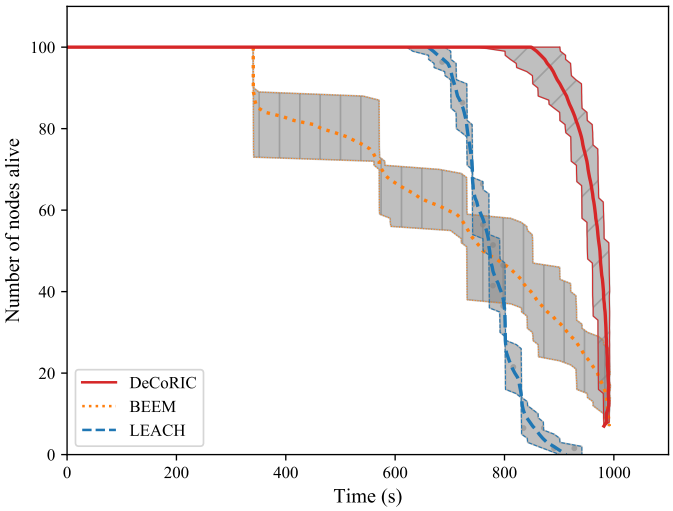

Fig. 7: Battery drain of the network of DeCoRIC, LEACH and BEEM. The gray area indicates the variation between the minimum and maximum boundaries with the solid line representing the average.

that start consuming energy after the death of some CH nodes, leading to a longer lifetime for these nodes. However, since nodes exhaust their energy at an early stage in BEEM, some key bridge nodes could exhaust energy quicker than the other nodes, leading to a disconnected network. DeCoRIC manages power more efficiently using RDC, providing a longer time for the network to stay connected before the nodes exhaust their powers. Since both DeCoRIC and BEEM aim to achieve connectivity, we see that the number of active nodes at the end of the experiment is similar for both algorithms.

B. Connectivity

We compare the clustering algorithms with respect to their ability to achieve connectivity among the nodes. In order to test connectivity among nodes of the network, we reduce the transceiver range to 20m, as larger transmission range in a denser network enables all the nodes to be in communication range with each other. This is a common strategy in dense networks for mitigating collisions, thereby reducing retransmissions [33]. Reduction in range enables reduction in transmission power which is the goal for most wireless sensors and IoT devices.

We measure the connectivity as the ratio of the number of connected nodes over the total N nodes in the network. Non-CH nodes of a cluster form a connected pair with its CH. Similarly, neighboring CH nodes form a connected path among the clusters. Combining such pairs, we get all the nodes that are connected in the network (where a routing path exists). Depending on the topology, the maximum connectivity could vary from a single cluster covering a few nodes to multiple clusters covering all N nodes of the network. The former is a result when CHs are not in range of each other, forming independent clusters, and the latter is formed when all CHs are in range of one another to form a path among all N nodes.

While DeCoRIC and BEEM strive to achieve 100% connectivity, DeCoRIC completes the clustering with less power

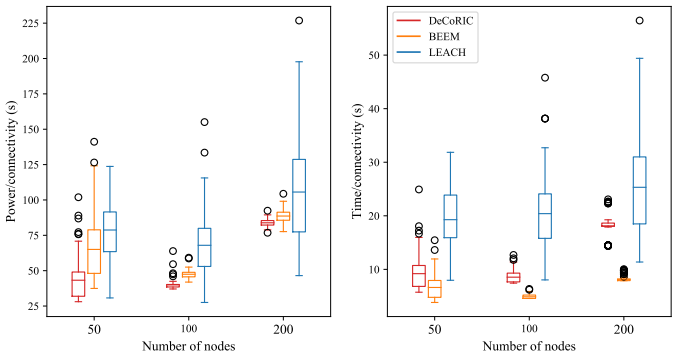

Fig. 8: Power consumed and time taken for the network clustering normalized by connectivity among all the nodes in the network.

and slightly longer time than BEEM. LEACH consumes the least time and power for clustering but does not ensure connectivity. Hence, in order to compare the performance of all the clustering schemes, we normalize both the power (power/connectivity) and time (time/connectivity) in the clustering phase by the connectivity achieved by the algorithms.

The results of the comparison are shown in Figure 8. The x-axis represents the number of nodes in the network. The y-axis of the left sub-plot represents the ratio of clustering power over connectivity while the y-axis of right sub-plot indicates the ratio of clustering time over connectivity. As seen from the left sub-plot, DeCoRIC expends the least power to achieve 100% connectivity, followed by BEEM and LEACH. In contrast to the clustering power, DeCoRIC needs slightly longer to complete clustering compared to BEEM as shown in the right sub-plot. The variations are attributed to the randomness of the topologies yielding different extents of connectivity.

As the number of nodes scales, the density of nodes increases, leading to better connectivity. Although LEACH consumes the least power and time to complete clustering, the results normalized over connectivity show that LEACH would need much higher time to move towards 100% connectivity. Additionally, the 100 random topologies are representative of the changes in connectivity due to the change of CHs resulting from re-clustering changes. BEEM also includes the cyclic re-clustering but retains the same CH to maintain connectivity. This property of BEEM expends significant energy, retaining connectivity only while the CH nodes are alive.

Similarly, the CHs are retained after the clustering is complete in DeCoRIC. However, over multiple epochs, the power consumption reduces significantly for DeCoRIC due to better radio management of the nodes. The longer time of clustering in DeCoRIC is attributed to the Correction phase where the number of CH nodes is reduced while striving to attain 100% connectivity. The slightly longer clustering of DeCoRIC creates optimal clusters that sustain the connectivity for a longer time. BEEM is faster as it does not consider the number of CH nodes active while achieving connectivity, thereby expending additional energy and leading to faster node deaths as seen in Figure 7. Hence, overall connected time for BEEM is significantly lower than DeCoRIC, where DeCoRIC achieves over 2x longer connected time.

TABLE III: Best and worst-case reaction time at each node with DeCoRIC in case of network changes.

Network change	**Detection time**		**Recovery time**
	Best-Case	Worst-Case	
Fail: nCH node	$2 \cdot T_{\text{fail}}^{\text{nCH}}$	$2.5 \cdot T_{\text{fail}}^{\text{nCH}}$	immediate
Fail: CH node	$2 \cdot T_{\text{fail}}^{\text{CH}}$	$2.5 \cdot T_{\text{fail}}^{\text{CH}}$	2 rounds
Fail: Bridge-CH node	$2 \cdot T_{\text{fail}}^{\text{CH}}$	$2.5 \cdot T_{\text{fail}}^{\text{CH}}$	2 rounds
Add: New node	3 rounds	1 cycle	[0 or 3] rounds

C. Evaluation of Resilience

DeCoRIC does not assume any synchronization across the nodes and the round/cycle period in DeCoRIC aims to compensate for the lack of synchronization between the nodes, as explained in Section III-B. Since each round specifies a periodic set of actions (i.e., CH transmission, non-CH nodes receiving without any sequence order), the timing drift between nodes can be bounded to one round. As explained in Section III-B3, the failure window (T_{fail}) covers the uncertainties caused due to the asynchronous RDC periods and transmission times, by defining T_{fail} as the least common multiple of the respective time periods.

If a transmission is not received at its neighbor and the neighbor receives a late gossip before the fail counter expires (at T_{fail}), then the counter is halved as it waits to see if it was a transient fault. Thus, in the worst-case, there can be an additional half period (of T_{fail}) that a neighbor waits before declaring the node to have failed. The fail period depends on the activity rate of the node; for CH nodes, their failure can be detected within a shorter window ($T_{\text{fail}}^{\text{CH}}$) compared to non-CH nodes, since CH nodes transmit more often than non-CH nodes. These bounds are thus enforced by the protocol and are shown in Table III. Once a failure of CH or Bridge-CH node is detected, the nodes switch to the Election phase immediately and complete the recovery process over the next 2 rounds. In the case of a non-CH node failure, the recovery is immediate as there are no changes triggered in the cluster itself.

When a new node integrates into the cluster, it can start following a CH within 3 rounds by listening to its broadcast messages. However, the new node can only determine its own degree over the next cycle when other non-CH nodes transmit. If the new node has a higher degree than the current CH, it will transition as the CH by setting the *New CH ID* field of the message frame, causing affiliated nodes to switch to the election phase to complete recovery. Otherwise, the recovery is completed immediately, and the new node integrates as a regular non-CH node. We verified the bounds stated in Table III using our simulations, starting with a stable network condition and random topologies.

Meanwhile, LEACH and BEEM do not have a failure detection mechanism within the protocol. Clustering operation repeats after every epoch, providing an upper bound for time to re-cluster as there is no failure detection mechanism. This property results in a constant recovery time for any CH node independent of the topology changes in the network. However, the worst-case recovery time could be longer depending on the configuration of an epoch.

V. Conclusions and Future Work

In this paper, we proposed a power-efficient decentralized clustering technique that can dynamically detect and adapt to node failures at runtime while ensuring connectivity among the nodes. The protocol enables identification of nodes which enable connectivity and strives to create clusters that are connected. DeCoRIC provides a resilient and reliable communication framework in a network of any topological structure. We show that the network can re-organize to form new clusters while maintaining connectivity even in case of critical CH failures, with a deterministic latency. We implemented the state-of-the-art benchmark clustering protocol LEACH as well as BEEM, the protocol for connectivity, in the Contiki simulator for comparison with DeCoRIC.

We demonstrate that the number of CHs that are elected independently in DeCoRIC is similar to the number of CHs decided apriori in centralized schemes. We also showed that DeCoRIC achieves up to 70% better power efficiency and 42% longer lifetime compared to LEACH while achieving up to 110% better power efficiency and 109% longer network lifetime in comparison to BEEM. Connectivity is achieved among nodes even in sparse networks using less power by accepting a slightly longer time for the clustering phase compared to BEEM.

In the future, we aim to further enhance the failure detection model in DeCoRIC and incorporate a time-synchronization framework which can aid in further improving energy consumption. We also aim to improve the initial clustering power consumption and minimize the detection time based on a more reliable physical layer protocol. Furthermore, we plan to deploy this technique on a hardware testbed as required in safety-critical IoT applications.

References

[1] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. Int. Conf. on System Sciences*, Jan. 2000.

[2] L. Xu, G. M. P. O'Hare, and R. Collier, "A balanced energy-efficient multihop clustering scheme for wireless sensor networks," in *7th IFIP Wireless and Mobile Networking Conference*, May 2014, pp. 1–8.

[3] O. Younis and S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks," *IEEE Trans. on Mobile Computing*, vol. 3, no. 4, pp. 366–379, Oct. 2004.

[4] S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, pp. 924–935, Sept. 2002.

[5] S. Yi, J. Heo, Y. Cho, and J. Hong, "Peach: Power-efficient and adaptive clustering hierarchy protocol for wireless sensor networks," *Computer Communications*, vol. 30, no. 14, pp. 2842 – 2852, 2007.

[6] Chengfa Li, Mao Ye, Guihai Chen, and Jie Wu, "An energy-efficient unequal clustering mechanism for wireless sensor networks," in *IEEE Int. Conf. on Mobile Adhoc and Sensor Systems Conference, 2005.*, Nov. 2005, pp. 8 pp.–604.

[7] E. Hartuv and R. Shamir, "A Clustering Algorithm Based on Graph Connectivity," *Information processing letters*, vol. 76, no. 4-6, 2000.

[8] O. M. Alia, Z. Shaaban, A. Basheer, A. Al-Ajouri, and A. Alsswey, "Musicians'-inspired clustering protocol for efficient energy Wireless Sensor Networks," in *Proc. Int. Conf. on Communications and Networking (ComNet)*, Mar. 2014, pp. 1–6.

[9] A. A. A. Ari, B. O. Yenke, N. Labraoui, I. Damakoa, and A. Gueroui, "A power efficient cluster-based routing algorithm for wireless sensor networks: Honeybees swarm intelligence based approach," *Journal of Network and Computer Applications*, vol. 69, pp. 77–97, 2016.

[10] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Trans. on Wireless Communications*, Oct. 2002.

[11] F. Avril, T. Bernard, A. Bui, and D. Sohier, "Clustering and communications scheduling in WSNs using mixed integer linear programming," *Journal of Communications and Networks*, vol. 16, pp. 421–429, 2014.

[12] R. S. Elhabyan and M. C. Yagoub, "PSO-HC: Particle swarm optimization protocol for hierarchical clustering in wireless sensor networks," in *Proc. Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, 2014, pp. 417–424.

[13] D. Karaboga, S. Okdem, and C. Ozturk, "Cluster based wireless sensor network routing using artificial bee colony algorithm," *Wireless Networks*, vol. 18, no. 7, pp. 847–860, 2012.

[14] V. Loscri, G. Morabito, and S. Marano, "A two-levels hierarchy for low-energy adaptive clustering hierarchy (TL-LEACH)," in *Proc. IEEE Vehicular Technology Conf. (VTC)*, vol. 3, 2005, pp. 1809–1813.

[15] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in *Proc. IEEE Joint Conf. on Computer and Communications (INFOCOMM)*, vol. 3, Mar. 2003, pp. 1713–1723 vol.3.

[16] T. M. Behera, S. K. Mohapatra, U. C. Samal, M. S. Khan, M. Daneshmand, and A. H. Gandomi, "Residual energy-based cluster-head selection in wsns for iot application," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5132–5139, June 2019.

[17] S. Al-Sodairi and R. Ouni, "Reliable and energy-efficient multi-hop leach-based clustering protocol for wireless sensor networks," *Sustainable Computing: Informatics and Systems*, vol. 20, pp. 1 – 13, 2018.

[18] L. Xu, R. Collier, and G. M. P. OHare, "A survey of clustering techniques in wsns and consideration of the challenges of applying such to 5g iot scenarios," *IEEE Internet of Things Journal*, pp. 1229–1249, Oct. 2017.

[19] M. Youssef, A. Youssef, and M. Younis, "Overlapping Multihop Clustering for Wireless Sensor Networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 20, no. 12, pp. 1844–1856, Dec. 2009.

[20] H. Chan and A. Perrig, "ACE: An Emergent Algorithm for Highly Uniform Cluster Formation," in *Proc. Wireless Sensor Networks*, 2004.

[21] X. Cai, Y. Duan, Y. He, J. Yang, and C. Li, "Bee-sensor-c: An energy-efficient and scalable multipath routing protocol for wireless sensor networks," *Int. Journal of Distributed Sensor Networks*, p. 976127, 2015.

[22] R. K. Jallu, P. R. Prasad, and G. K. Das, "Distributed construction of connected dominating set in unit disk graphs," *Journal of Parallel and Distributed Computing*, vol. 104, 2017.

[23] X. Gao, X. Zhu, J. Li, F. Wu, G. Chen, D. Du, and S. Tang, "A novel approximation for multi-hop connected clustering problem in wireless networks," *IEEE/ACM Tran. on Networking*, vol. 25, Aug 2017.

[24] T. Shi, S. Cheng, Z. Cai, and J. Li, "Adaptive connected dominating set discovering algorithm in energy-harvest sensor networks," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[25] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)," *IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003)*, Sep. 2006.

[26] L. Xu, M. J. O'Grady, G. M. P. O'Hare, and R. Collier, "Reliable multihop intra-cluster communication for wireless sensor networks," in *2014 Int. Conf. on Computing, Networking and Communications (ICNC)*, Feb. 2014, pp. 858–863.

[27] A. N. Alvi, S. H. Bouk, S. H. Ahmed, and M. A. Yaqub, "Influence of Backoff Period in Slotted CSMA/CA of IEEE 802.15.4," in *Int. Conf. on Wired/Wireless Internet Communication*, May. 2016, pp. 40–51.

[28] H.-W. Tseng, Y.-C. Fan, S.-T. Sheu, and S.-Y. Ou, "An effective grouping scheme for avoiding hidden node problem in ieee 802.15.4-based wireless sensor networks," *SIGAPP Appl. Comput. Rev.*, vol. 14, pp. 30–40, Mar. 2014.

[29] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proc. of the IEEE Int. Conf. on Local Computer Networks*, 2004, pp. 455–462.

[30] R. van Renesse, Y. Minsky, and M. Hayden, "A Gossip-Style Failure Detection Service," in *Proc. Middleware*, 1998.

[31] Moteiv Corporation, "Tmote sky," https://insense.cs.st-andrews.ac.uk/files/2013/04/tmote-sky-datasheet.pdf.

[32] A. Riker, M. Curado, and E. Monteiro, "Neutral operation of the minimum energy node in energy-harvesting environments," in *IEEE Symposium on Computers and Communication (ISCC)*, July 2017.

[33] J. Shin, U. Ramachandran, and M. Ammar, "On improving the reliability of packet delivery in dense wireless sensor networks," in *16th Int. Conf. on Computer Communications and Networks*, Aug. 2007, pp. 718–723.