Abstract— In this work we design a virtual backbone to provide energy-efficient communication in Wireless Sensor Networks (WSNs). WSNs need virtual backbones to support in-network data transmission. In scenarios where there is no spatial redundancy (such as a sensor network in a petroleum installation), Connected Dominating Set (CDS)-based algorithms can be useful for hierarchical network management. We construct a CDS-based backbone to support the operation of an energy efficient network. We focus on three key ideas in our design: (a) a realistic weight matrix, (b) an asymmetric communication link between pairs of nodes, and (c) a role switching technique to prolong the lifetime of the CDS backbone. We simulate our proposed design using realistic scenarios for performance evaluation. We compare the proposed design with a traditional CDS algorithm. The simulation results prove the efficiency of the proposed algorithm in terms of network lifetime and packet loss.

Key words: Wireless Sensor Network; Connected Dominating Set; Virtual Backbone; Network lifetime.

I. INTRODUCTION

A Wireless Sensor Network (WSN) includes deployment of sensor nodes for monitoring environment. Sensor nodes are generally autonomous and communicate to each other via wireless media. The deployment of sensors can be dense and as large as 20000 nodes (in an industrial automation). In order to manage network traffic, WSN needs some form of virtual backbone. We can use the notion of connected dominating set (CDS) to create virtual backbone in WSNs. In CDS-based backbone, fewer nodes need to keep track of routing information; these nodes are part of Dominating Set (DS). Thus, relatively less number of nodes receive routing control messages in case of any topological change. A CDS can also reduce the impact of the broadcast storm problem [1]. Furthermore, using CDS backbone, we can put non-CDS nodes into periodical sleep mode to conserve overall energy of the network [2].

In this work, we use CDS-based network backbone for a sensor network and try to extend the network life time. From a systems perspective, we consider realistic sensor network characteristics to implement and analyze energy-efficiency for the WSN. The remaining of this paper is organized as follows: in Section II, we state the problem that we address in this work. Section III discusses some related works and then formulates the motivation of our work. Later on, in Section III-B, we describe the details of \( r \)-CDS algorithm proposed by Li et al. [3]. Section IV provides the motivations of our work. Our extensions to \( r \)-CDS algorithm are presented in Section V. Section VI-E describes results and analysis. Finally, Section VII concludes the paper.

II. PROBLEM DEFINITION

Any computing network needs to have some form of backbone to support in-network data traffic. WSN is not an exception, but unfortunately in most cases WSNs do not have any fixed network structure. Nodes may run out of battery power or they may get physically moved, resulting in changes in network topology. Hence, it requires some form of virtual backbone that can be formed on-the-fly and yet support data flow.

There are many algorithms to construct a virtual backbone using CDS. Most of those algorithms mainly focus on two aspects: (a) reducing backbone size; and (b) reducing time/message complexity. There is a need for research that can design CDS backbone for extending network lifetime.

In case of many industrial WSNs, every data packet is important. Hence, for such a network, the network life effectively ends with the First Node Death (FND). Due to incomplete data, the remaining energy in the surviving nodes is of no use after FND. Keeping this in mind, our aim in this work is to design a CDS backbone that can delay the death of the first node. We try to prolong network lifetime and reduce energy wastage.

III. LITERATURE REVIEW

A. Previous Works

There has been many research works done on constructing a CDS-based backbone in WSN. Based on the information required to build a CDS, we can divide the existing algorithms into 2 broad categories: (a) Centralized, and (b) Distributed. Further, if we consider the certainty of a CDS construction, algorithms fall into either: (a) Probabilistic, or (b) Deterministic.

As centralized CDS construction algorithms require global information on the network, these approaches are not well
suited for: (a) large networks due to a massive number of nodes; (b) dynamic networks due to a variable number of nodes. However, distributed algorithms only need \( n \)-hop neighborhood information for small \( n \). These algorithms have low construction cost and show fast convergence.

Probabilistic techniques incur low overhead and create CDS with high probability [4]; however, these are network dependent. For a dense network, the backbone size becomes large in these approaches. Deterministic schemes keep the backbone small but require high computation [5]. The good part of a deterministic approach is that it guarantees a CDS in connected networks [6].

Guha et al. proposed a centralized algorithm in their seminal work [7] on CDS. Their algorithm starts from a single node and later on, iteratively, more nodes are added to it. At first all nodes are marked white and the node with maximum white neighbors initiates CDS construction. That node turns into black and its white neighbors turn into gray. Then, the algorithm iteratively checks gray neighbors of it as well as all gray-white pairs, for a candidate having maximum white neighbors. Then, that node/pair of nodes becomes black. In this way, the algorithm continues until there is a single white node in the network.

Das et al. [8] proposed a centralized algorithm to construct CDS in a WSN. According to their design, each node needs to know if it has the maximum degree among all nodes in the network. It also requires the backbone nodes to know which 1- and 2-hop-away neighbors are marked. These two requirements force the flooding of node-degree information in the network.

Li et al. [3] designed a distributed localized algorithm (r-CDS) to construct a 1-CDS. At first this algorithm chooses a set of Dominators for the backbone, and then, as the CDS construction progresses, some connector nodes get selected to connect the initial Dominators. Similar to other algorithms, it also uses some pruning techniques to reduce the number of dominator nodes in the network backbone. As we base our work on this particular algorithm, we present further discussion on this in Section III-B.

Wu et al. [9] proposed both centralized and distributed algorithms to construct k-connected m-dominating set (km-CDS) backbones for WSNs. Their distributed km-CDS construction is based on Li et al.’s r-CDS algorithm. Earlier works to build CDS focused on special cases like \( k=1 \), 2 or \( k=m \) but Wu et al. designed algorithms for general \( k \) and \( m \). Both algorithms presented in this work form a km-CDS in two phases. In the first phase, algorithms form an m-dominating set, and in the second phase they ensure k-connectivity for that set.

### B. Details of the r-CDS Algorithm

In our work, we modify the r-CDS algorithm proposed by Li et al.. We choose to use this algorithm because it is distributed in nature, and hence does not require global information. At the same time, it is deterministic. Moreover, it has already been used to construct km-CDS [9] for fault tolerance, which reflects its extendability. In this section, we describe the details of this algorithm.

r-CDS algorithm constructs a 1-CDS in a WSN. Based on the weight comparison among neighbors, some suitable nodes get selected as Dominators. The set of Dominators is a MIS. Those initially selected Dominators, in conjunction with some Connector nodes (known as Dominatee2 nodes), later on form the dominating set (or backbone) of the network. On the other hand, nodes that are not part of the dominating set remain as Dominatees, and use neighboring Dominators as next hops for data communication. This algorithm assumes that all nodes know 2-hop-away neighborhood information and have equal transmission range. Table I defines different notations used in this algorithm.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d(u) )</td>
<td>Degree of node ( u )</td>
</tr>
<tr>
<td>( \deg(u) )</td>
<td>The effective node degree of node ( u ). It is the count of white neighbors of ( u )</td>
</tr>
<tr>
<td>( r(u) )</td>
<td>The number of 2-hop away neighbors - ( d(u) ); ( r ) for brevity</td>
</tr>
<tr>
<td>( N(u) )</td>
<td>Node set that includes ( u )’s all 1-hop neighbors excluding ( u ), also called open neighbor set of node ( u )</td>
</tr>
<tr>
<td>( N[u] )</td>
<td>( N(u) \cup {u} ), also called the close neighbor set of node ( u )</td>
</tr>
</tbody>
</table>

1) **Weight Comparison**: The weight matrix used in r-CDS algorithm is: \( W_i(r_{ij}, \deg_i, ID_i) \). During weight comparison, node \( i \) is more suitable to be a Dominator than neighboring node \( j \), if any of the following is true:

1. \( r(i) < r(j) \)
2. \( r(i) = r(j) \) and \( \deg(i) > \deg(j) \)
3. \( r(i) = r(j) \) and \( \deg(i) = \deg(j) \) and \( ID(i) < ID(j) \)

2) **Colors and Messages**: Sensor nodes in the r-CDS algorithm can have three different colors: white, gray and black. Initially all nodes are white. During the execution of algorithm, all nodes change their color to either black or gray. Black nodes form network backbone, whereas gray nodes remain as Dominatees. In this algorithm, nodes can broadcast the following messages:

1. BLACK Message: If a node turns into a Dominator node, it sends out this message.
2. GRAY Message: If a node turns gray, it sends out this message. The GRAY message contains two IDs: its own node ID and the ID of the neighbor that caused it to turn gray.
3. \( \deg(u) \) Message: When a white node receives GRAY or BLACK message, its effective node degree changes, and hence it broadcasts this message.

3) **Selecting Dominator1 Nodes**: After each node knows about its 2-hop-away neighborhood by neighbor discovery, all nodes broadcast their \( r \) values. A node \( u \) becomes Dominator1 if it wins in the weight comparison over its neighbors. Then, node \( u \) turns black and broadcasts a BLACK message in the neighborhood. If a white node \( v \) receives BLACK message from its neighbor \( u \), \( v \) becomes gray and broadcasts GRAY
message in its neighborhood. This GRAY message contains the pair (v’s ID, u’s ID).

4) Selecting Dominator2 Nodes: If a black node w receives GRAY message from a gray node v and the ID of another black node u, and if w and u are not connected yet, then v becomes Dominator2 node (and hence turns into black) in order to connect u and w. In the same way, after receiving a BLACK message from a node w, if a gray node u has already received a notification that there is a 2-hop-away black neighbor v sent by a neighbor x and v has not been connected to w yet, then both u and x become Dominator2 nodes to connect node v and node w. Upon receiving a GRAY or a BLACK message, a white node u decrements its effective degree $deg(u)$ by 1 and broadcasts $deg(u)$ message.

IV. MOTIVATIONS OF OUR WORK

Every algorithm proposed so far uses some form of weight matrix to pick dominator nodes. Table II lists some algorithms with respective weight matrices. For each weight matrix, the ordering of elements is important. For example, according to the algorithm presented in [10], $W_i(N_i, ID_i) > W_j(N_j, ID_j)$ if either of the followings holds:

- $N_i > N_j$ or
- $N_i = N_j$ and $ID_i > ID_j$

In Table II, we see that some CDS algorithms do not consider the remaining energy of the node, which is unrealistic. Again some other algorithms consider node degree to be the prime factor in weight matrix. According to those algorithms, the higher the node degree, the more is its chance to become dominator. Putting prime importance on one parameter during tuple-based comparison can go either way: small dominator set (when degree is first parameter) with low-energy; or large dominator set (when remaining energy is first parameter). We propose a realistic, composite weight comparison in section V-A.

Further, almost all algorithms consider bidirectional communication across any edge in the graph. It is assumed that if node A can reach node B, node B will also be able to reach node A. Thus they consider uniform transmission range throughout the network that resembles an unit disk graph. However, real life sensor network can have sensor node with variable transmission range, causing asymmetric communication across a link. In section V-B, we explain how we address this issue.

In general, algorithms used to design CDS backbone do not consider switching the role of dominators. But if we use same set of dominators, those nodes will carry or relay network traffic all the time. As a result, they will run out of energy quite early compared to dominator nodes. So research works that aim at delaying the death of first node, switching role of dominators can be useful. For example, once the energy of a dominator goes below a predefined threshold, a different node can take up the responsibility. In section V-C, we discuss to incorporate role switching of dominators.

V. EXTENSION OF r-CDS ALGORITHM FOR ENERGY-EFFICIENCY

In this section, we propose our modifications to r-CDS algorithm in order to prolong network lifetime. With the modifications we propose, we name the algorithm ‘modified r-CDS’ or ‘$m_r$-CDS’.

A. Weight Comparison

In our algorithm, we propose to use a composite weight matrix comparison instead of single-element tuple-based comparison. We propose to reduce the weight matrix of the r-CDS algorithm (Section III-B.1) into $W_i(x_i, ID_i)$, where $x_i = f(N_i, e_i)$. Similar to the weight comparison of the r-CDS, the following weight computation and comparison are executed on each node of the network. We assume that during network operation, all nodes send same-length data packets in each data gathering round. Our target here is to delay the first node death. Table III presents different terms that we use in our estimation of remaining energy during weight computation.

| Table II |

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Weight matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>[5]</td>
<td>${S(i, t), \text{ for k-coverage}}$</td>
</tr>
<tr>
<td>[9]</td>
<td>$(N_i, e_i, ID_i)$</td>
</tr>
<tr>
<td>[10]</td>
<td>$(N_i, ID_i)$</td>
</tr>
<tr>
<td>[12]</td>
<td>$(e_i, N_i, ID_i)$</td>
</tr>
<tr>
<td>[13]</td>
<td>$(ID_i, e_i, M_i, T_i)$</td>
</tr>
<tr>
<td>[14]</td>
<td>$N_i$</td>
</tr>
</tbody>
</table>

$ID_i$: node id for node i
$N_i$: node degree of node i
$e_i$: remaining energy of node i
$S(i, t)$: forwarding status of node i
$M_i$: mobility of node i
$T_i$: traffic load of node i

| TABLE III |

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_{tx}^u$</td>
<td>Energy cost to send 1 packet from node i (in Joule)</td>
</tr>
<tr>
<td>$e_{rx}^u$</td>
<td>Energy cost to receive 1 packet at node i (in Joule)</td>
</tr>
<tr>
<td>$e_i$</td>
<td>Initial energy of node i (in Joule)</td>
</tr>
<tr>
<td>$N_i$</td>
<td>Node degree of node i</td>
</tr>
<tr>
<td>$m$</td>
<td>Initial data round count that network is expected to operate</td>
</tr>
<tr>
<td>$p_i^m$</td>
<td>Total packet reception at node i after k data rounds</td>
</tr>
</tbody>
</table>

1) Estimating Remaining Energy: Consider the following:

\[ p_u^m = N_i \times m \]
\[ p_v^m = N_i \times m \]

\[ drain_u = m \times e_{tx}^u + p_u^m \times (e_{rx}^u + e_{tx}^u) \]
\[ drain_v = m \times e_{tx}^v + p_v^m \times (e_{rx}^v + e_{tx}^v) \]

After $m$ rounds,
The total energy drain at any node is the sum of its own packet transmission cost and the relay cost.

2) Selection of Dominator: Algorithm 1 provides the details of selection of a dominator by comparing the node weights.

Algorithm 1 Algorithm to Compare Node Weights in mr-CDS
/* $x_u$ and $x_v$ are remaining energy of node $u$ and $v$ after $m$ rounds */
if $x_u > x_v$ then
    $u$ becomes Dominator
else
    if $x_u < x_v$ then
        $v$ becomes Dominator
    else
        if $ID_u < ID_v$ then
            $u$ becomes Dominator
        else
            $v$ becomes Dominator
    end if
end if

B. Variable Transmission Range

During neighbor discovery phase of the $r$-CDS algorithm, each node broadcasts hello packet. Nodes that receive any such packet are confirmed neighbors of the hello sender. This is only possible if all communication links are symmetric. However, as we consider variable transmission range in the network, we cannot assume symmetric communication across any link. This brings up the need to apply acknowledgment scheme during neighbor discovery phase of the mr-CDS algorithm.

The 3-node network in Figure 1 explains link asymmetry in any WSN. In this Figure, $tr_w > tr_v > tr_u$, where $tr_i$ is transmission range of node $i$. Following the $r$-CDS algorithm, if all nodes broadcast hello packets in this configuration, node $v$ will receive hello packets from $u$ and $w$. But due to lower transmission range, $v$’s own hello packet will not reach $w$. So, even though $v$ receives hello packet from node $w$, bidirectional communication is not possible between them. In other words, they are not confirmed neighbor. In order to address this issue, we propose following modifications to the neighbor discovery phase of the $r$-CDS algorithm:

1) Each node sends out its hello packet mentioning its node ID, hop-count and remaining energy.

2) As nodes receive hello packets, each node maintains two lists for its potential neighbors: High-list and Low-list. If the remaining energy of the hello sender is greater or equal to its own energy, the hello sender is added to the High-list. On the other hand, a hello sender with lower energy ends up in the Low-list. For example, in Figure 1, after receiving hello packets, node $v$ puts $u$ in Low-list and $w$ in High-list. Now, the node in Low-list is its confirmed neighbor, as with greater transmission power, $v$ can reach it. For the node in the Low-list, it sends out an ack packet. And for the node in the High-list, it waits to listen for an ack packets in response to its hello packet.

3) After listening for a predefined period of time, if a node does not receive any ack packet from some node in its High-list, it removes those nodes from the list. For example, in Figure 1, as the hello packet from node $v$ can not reach $w$, $w$ will not send any ack packet to $v$. But $v$ will send an ack packet to $u$. In the end, nodes of both lists make up a node’s neighborhood.

C. Role Switching Technique

In Section IV, we stated the reason for role switching of the Dominators. It can help delay the death of the first node in a network. In order to delay the first-node death, we propose the following rules for role switching:

Rule 1: Consider a Dominator node $k$ and the following conditions are true:
- $k$ is one hop away from sink,
- $k$’s remaining energy is one-third of its initial energy, and
- $k$ has $n$ number of neighbor Dominators, $j_1 \ldots j_n$, for which it forwards their packets to the sink.

Then, $k$ requests $\forall j$ to increase their transmission levels so that they can directly send their packets to the sink.

![Fig. 1. Neighbor Discovery in Asymmetric Communication](image)
Rule 2: Consider a Dominator node \( j_i \) which receives a request from another Dominator \( k \) to increase its \((j_i)\)'s transmission power level. The Dominator \( j_i \) will act as follows:

- If \( j_i \) can directly reach the sink by increasing its current transmission power, \( j_i \) increases the transmission power level and transmits subsequent packets directly to the sink instead of current next hop \( k \). The sink acknowledges the packet sent because of increasing power level and the sink will add \( j_i \) in it’s neighbor list.
- Otherwise, if \( j_i \) cannot directly transmit to the sink even with its maximum transmission power, \( j_i \) ignores the current request and continue transmitting packets via \( k \).

VI. RESULTS AND DISCUSSION

We use Java-based custom simulator to evaluate the performance of both \( r \)-CDS and \( mr \)-CDS algorithms. The aim of different simulation experiments is to compare \( mr \)-CDS with \( r \)-CDS in terms of network lifetime and packet loss. Besides, we analyze energy consumption for both algorithms. We also investigate on how both algorithms perform under varying conditions like- Node Density and Average Node Energy.

A. Simulation Terms

During the energy cost analysis, we use the following terms:

1) Alive Node: A node is considered alive if it can send out a packet to next hop towards the sink. In other words, a node is alive as long as its remaining energy is greater than the energy required to transmit a single packet.

2) Data Round: Each data round is defined by all alive nodes sending out a single packet each, to the sink node.

3) Lost packet: A packet is considered lost if it leaves source node but can not reach the sink.

4) Dead Node Count (DNC) at Round \( i \): Global metric to count the number of dead nodes at round \( i \).

5) Packet Loss Count (PLC) at Round \( i \): Global metric to count the number of packets lost till the end of data gathering round \( i \). For example, let us consider only node \( x \) dies in data round \( i \) (i.e., DNC=1). After \( x \)'s death, if some other nodes (for which \( x \) worked as next hop towards sink) sends \( \Delta \) packets during round \( i \), all those packets will be lost. Then we can say PLC at round \( i \),

\[ PLC_i = PLC_{i-1} + \Delta. \]

B. Radio Model

In our simulation, we use First Order Radio Model described in [15]. During data reception, a node drains energy for running the radio circuit only. But for data transmission, it also drains energy to amplify the signal. The energy required to run the radio circuit (at transmitter/receiver) is \( e_{elec} = 50nJ/bit \). And energy required to run the amplifier during transmission is \( e_{amp} = 100pJ/bit/m^2 \). So if node \( i \) tries to send a \( k \)-bit packet to node \( j \), which is \( d_{ij} \) meters away, energy consumption at node \( i \) is:

\[ T_{ij} = k.e_{elec} + k.e_{amp}.d_{ij}^2. \]

On the other hand, any node \( i \) that receives a \( k \)-bit packet drains only \( R_i = k.e_{elec} \) amount of energy.

C. Network Model

In our simulation, we consider 100 sensor nodes and a single sink node deployed at random in an area of 50m×50m. Transmission ranges of these nodes vary between 20 to 30 meters. In order to get reliable results, we consider 20 such different networks and take average of them to get all statistics. During each data gathering round, we consider that each alive node sends out a 1000-bit data packet. We consider nodes are stationary, i.e., once deployed, they do not move. Further, similar to the works done in [16] and [17], wireless channel is assumed to be ideal where there is no retransmission of packets due to collision. We consider that constructing a CDS backbone is a one-time operation and hence, we do not add the energy cost to construct CDS backbone in our energy consumption analysis.

D. Routing Algorithm

After running \( r \)-CDS or \( mr \)-CDS algorithm, each node in the network knows about its Dominator neighbor(s). But other than the nodes that have single Dominator in their neighborhood, no node has any idea on what is the next hop Dominator in a least cost path towards the sink. In order to solve this problem, we need a data collection tree after forming a CDS-based backbone. We apply Distance Vector (DV) Routing Algorithm [18] to construct such a tree. The base station acts as the root node and initiates tree formation. Only the Dominator nodes take part into the construction of this least-cost path tree. We can see in Table VI that the number of Dominator nodes is quite low compared to the network size, so running DV algorithm over only Dominators reduces the size of the input space.

E. Results

We present simulation results in this section. It also contains our analysis on the observed results.

1) Network Lifetime and Packet Loss Count: We evaluate the performance of both implementations (i.e., \( r \)-CDS and \( mr \)-CDS) in terms of Dead Node Count (DNC) and Packet Loss Count (PLC). Nodes were randomly assigned energy of 2 to 3 Jouls. With a total of 100 nodes in our simulation, Table IV presents data for the first 10 node deaths (which is one-tenth of the network) using \( r \)-CDS algorithm. Here we can see that in average (over 20 simulation networks), first node death occurs in data round 591. Besides, in average, 4 packets are lost by the end of the data round during which the first node death occurred.

Similarly, Table V shows node deaths and PLCs for the \( mr \)-CDS algorithm. If we compare Table IV with Table V, we can see that our \( mr \)-CDS algorithm significantly delays the first node death and is far better in terms of PLC. On an average, the first node death occurs in data round 1,189 in the \( mr \)-CDS, which is more than the double of the original \( r \)-CDS network lifetime. A comparison of node deaths is portrayed in Figure 2 which also reveals a critical issue: using \( mr \)-CDS we get a gradual pattern in node deaths.
After the crossover point between $r$-CDS and $mr$-CDS in Figure 2, the average $r$-CDS line is steep (yet gradual) for a while. Later on, that line is almost constant, which means all nodes start to die at the same time. On the other hand, $mr$-CDS nodes die gradually after the crossover point, and this gradual trend continues for almost more than the double of gradual node deaths in the $r$-CDS. This pattern of node death reflects load balancing of the network in the $mr$-CDS algorithm and is due to the role switching scheme adopted in our design.

![Fig. 2. Comparison on Node Deaths](image)

In terms of packet loss, $mr$-CDS works far better than original $r$-CDS. This is due to the fact that some nodes in the $r$-CDS die quite early, causing a whole lot of packet loss early in the network operation, as shown in Table IV, Table V and Figure 3.

2) **Energy Efficiency in the $mr$-CDS Algorithm:**

![Fig. 3. Comparison on Packet Loss Count](image)

a) **Comparison on Backbone Size:** If we compare the size of network backbone, we can see that $mr$-CDS has little edge over $r$-CDS. Table VI shows the total number of Dominator1 and Dominator2 nodes in both $r$-CDS and $mr$-CDS backbones for our 20 input networks. On an average, 22.15 nodes constitute the $r$-CDS backbone, whereas in case of $mr$-CDS, we have 20.85 nodes in the backbone. As backbone nodes must remain active throughout network operation, with fewer nodes in the backbone, the $mr$-CDS algorithm consumes less overall energy to keep the network operational. Further, a smaller backbone in the $mr$-CDS can allow more nodes(non-Dominators) to be put in a periodic sleep mode and save network energy.

b) **Comparison on Remaining Energy of the Network after First Node Death:** For many data-critical applications, a sensor network is of no use once we stop receiving quality information out of it. So if the network lifetime ($FND$) expires, remaining energy of other alive nodes does not come to any use. An energy efficient design should try to maximize the use of overall network energy before a network expires. In this context, we examine the average amount of remaining energy on each node after $FND$. Table VII presents data for both $r$-CDS and $mr$-CDS. As stated in Section VI-E, nodes in our 20 input networks have energy between 2 to 3 Jouls, which averages to 2.136 Joul in our simulation. As we can see in Table VII, we have less energy waste in the $mr$-CDS algorithm compared to the $r$-CDS.

![Fig. 3. Comparison on Packet Loss Count](image)

![TABLE VII](image)

<table>
<thead>
<tr>
<th>Technique</th>
<th>Remaining Energy(J)</th>
<th>Energy Waste(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$-CDS</td>
<td>2.065</td>
<td>96.71</td>
</tr>
<tr>
<td>$mr$-CDS</td>
<td>1.989</td>
<td>93.16</td>
</tr>
</tbody>
</table>

Less energy waste in the $mr$-CDS algorithm is due to the fact that using this approach, $FND$ occurs later than $r$-CDS. The role switching property of $mr$-CDS does load balancing and as a result, we get less energy waste.

**VII. CONCLUSION**

In this work we extended an existing CDS algorithm ($r$-CDS) to meet real life sensor network requirements. We tried to extend network lifetime by delaying First Node Death. This
delay can help data-critical WSNs like an industrial WSN to continue network traffic for longer period of time. Simulation results proved that mr-CDS algorithm outperforms original r-CDS algorithm and significantly prolongs network lifetime. Our future focus is on implementing mr-CDS algorithm in real sensor hardware by using TinyOS/NesC programming.

VIII. ACKNOWLEDGEMENT

This work is partially supported by Dr. Sajid Hussain’s collaborative research contract with PAWS/CBU project funded by ACOA/AIF, Sydney, Canada and NSERC’s Discovery grant. Further, the work is partially supported by Dr. Lei Qian’s research funding with NOAA, USA.

REFERENCES


TABLE VI

<table>
<thead>
<tr>
<th>Experiment No.</th>
<th>r-CDS Backbone Node Count</th>
<th>mr-CDS Backbone Node Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dominator 1</td>
<td>Dominator 2</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>20</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>17</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>18</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>19</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>20</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>Average</td>
<td>8.25</td>
<td>13.9</td>
</tr>
</tbody>
</table>