J. Basic. Appl. Sci. Res., **2(12)12809-12821, 2012**
© 2012, TextRoad Publication

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

Global ID Assignment in Wireless Sensor Network by Using Huffman Algorithm

Parisa Mohammadi*[1], Dr Shahram Jamali[2]

[1]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Ardabil, Iran.
[2]Department of Computer Engineering, University of Mohaghegh Ardabili, Ardabil, Iran.

ABSTRACT

A sensor network consists of a set of battery-powered nodes, which collaborate to perform sensing tasks in given environment. It may contain node or more base stations to collect sensed data and possibly relay it a central processing and storage system. These networks are characterized by scarcity of resources, in particular the available energy. We present a Huffman algorithm to solve the unique ID assignment problem. The propos is based on Huffman codes as unique and short ID for nodes. As we know codes generated using this algorithm has some characteristics that makes it a good choice for addressing (prefix free, compressed) and some characteristics that makes it not to be considered as perfect choice (the length of these codes are not fixed). In this paper we have program c++ to simulate an area with a large number of sensors and put our desired way of addressing in it (using Huffman codes). And organizing nodes in a tree structure. This tree structure is used to compute the size of the network. Then we make a Huffman tree for unique Ids are assigned using the minimum number of bites. And evaluate its performance and compare results by tree cast method.The oure result show weak signal makes the network's tree deeper and the Huffman method is less bits needed for ID assignment rather the Tree Cast method and the depth of tree doesn't have a big influence on the length of IDs in the Huffman method.Globally unique IDs are useful in providing many network functions, e.g. configuration, monitoring or individual nodes, and various security mechanisms.

KEY WORDS: ID Assignment, Sensor networks, Huffman algorithm

1. INTRODUCTION

The communication range of individual sensor nodes is generally limited, and communication is often carried out in a multi-hop manner. There is a need to have a unique identifier in the header of every unicast packet. In fact, routing protocols need to uniquely identify the final destination as any node in the network can be a potential destination. Several routing protocols use attribute-based routing and therefore can use attributes as global identifiers. However, even these protocols require the existence of unique Ids at a local level. This is the case for directed diffusion [1] and geographical routing protocols such as [2]. Network-wide unique Ids are beneficial for administrative tasks requiring reliability, such as configuration and monitoring of individual nodes, and download of binary code or data aggregation descriptions to sensor nodes [3] network-wide unique Ids are also required when security is needed in sensor network [4]. Several MAC protocols requiring the pre-existence of network-wide unique IDs have also been proposed for sensor network [5]. Assumption of the pre-existence of network-wide IDs in not realistic in the case of sensor networks. The pre-existence of network-wide global IDs requires hard-coding these IDs on nodes prior to the deployment. This is costly in terms of time and effort when a network contains thousands to hundreds of thousands of nodes. Another alternative is to have MAC addresses that are unique for every manufactured sensor node, as is the case for Ethernet cards [6].This is not a desirable approach because of coordination it requires and the fact these IDs would have to be lengthy and therefore costly to use in packet headers. An obvious ID assignment strategy is to have each node randomly choose an ID such that probability of any two nodes choosing the same ID is very low. However, for this probability to be low, we need the IDs to be very long, which is again costly in terms of energy [7]. Any ID assignment solution should produce the shortest possible addresses because sensor networks are energy-constrained. The usage of the minimum number of bites required is motivated by the need to limit the size of transmitted packets, in particular the header. In fact, communication is usually the main source of energy drain in a sensor node [8]. We also do not assume the existence of any specific communication protocol. In this paper related work is discussed in section 2. In section 3 introduce Huffman algorithm for unique is assignment ,in section 4 performance evaluation , in section 5 comparison tree cast method with Huffman method , in section 5 include simulation results, in section 6 include weak points of our proposed algorithm and section 7 include conclusion.

2. Related Works

In general, network-wide unique addresses are not needed to identify the destination node of a specific packet in sensor networks. In fact, attribute-based addressing fits better with the specificities of sensor networks [9]. In this case, an attribute such as node location and sensor type is used to identify the final destination. However. Different nodes can have the same attribute value. in particular in the same neighborhood. Thus. There is a need to uniquely identify the next hop node during packet routing [10]. Furthermore. It is possible that two neighboring nodes have the same attributes. For instance, it is likely that some nodes will

***Corresponding Author:** Parisa Mohammadi, Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Ardabil, Iran. E.mail: Mohammadi_PM@yahoo.com, Tel: (++98)451-3367346

have the same location in a dense sensor network. In addition. The number of bits required to represent attribute information (for example the node geographical coordinates) may be large rendering this approach less attractive from a communication energy point of view [11].In [12]. The authors propose an algorithm that assigns globally unique IDs. Like our algorithm, it uses a tree structure to guarantee the uniqueness of each ID. The algorithm is similar to the first phase of our algorithm. It starts with the sink node broadcasting a message that contains its ID and a parameter b given the size in bits of one-hop ID. Successive nodes choose a parent node among their neighbors that already have an ID. The node then randomly chooses an ID of size b bits and relays on its parent to guarantee no other node has chosen the same ID. The node then appends its chosen ID to the ID of its parent to create a unique ID. The main difference between this algorithm and ours is that it does not use the network size to minimize the size of node IDs. Our algorithm. in contrast. Not only assigns unique IDs but also guarantees that these IDs are of minimum length. This is a considerable advantage considering that sensor networks are energy-sensitive. the authors propose a local ID assignment scheme where address conflicts are resolved in a Reactive way. Nodes randomly choose an address that is likely to be unique within a 2-hop neighborhood. No conflict resolution is performed until nodes need to enter in a communication. For instance, interest broadcasting in directed diffusion can be used to resolve conflicts. In this case, the sink node discovers the existence of identical IDs between its immediate neighbors. The conflicting nodes are notified and choose new IDs. Each node that forwards the interest message uses the response messages to detect ID conflicts among its neighbors. The delaying can help save energy by avoiding any unnecessary conflicted solution. In particular, if two neighbors have the choose the same local ID but are never active at the same time, resolving such a conflict amounts to a waste of energy resources. However, resolving ID conflicts reactively can be problematic if the sensor network requires time-sensitive exchange of information, since messages can be delayed to resolve an ID conflict. In [13]. The initialization has the objective of transitioning from this unstructured state to a structured network and the establishment of a MAC protocol to allow efficient information dissemination. An important way to structure sensor nodes into a network is through the use of clustering techniques [16, 15, 14]. Many of the clustering algorithms that have been proposed for sensor networks assume the pre-existence of unique node IDs. For example, in the approach proposed in [14] a newly active node that wants to join the network waits for messages from neighboring dominators (cluster heads) before trying to become a dominator. A unique ID is needed to distinguish between the different dominators in the neighborhood. The ID assignment algorithm proposed here can be used as part of the overall network initialization phase. In this way, the initialization does not need to require the pre-existence of node IDs. Like the work in [14], our approach does not assume that all nodes wake-up at the same time. We also do not assume the existence of a specific collision avoidance protocol. As "*Santashil PalChaudhuri*" described in "*Tree Cast [17]: A global addressing and Stateless Routing Architecture for Sensor Networks*" article, In this scheme nodes are organized in a tree structure and are assigned addresses according to their position in the tree. Sink node acts as the root tree. Child nodes parent node assign addresses to its descendant children. Child node randomly generates its own address notifies the conflicting the child node. If the generated local address of a node is approved by the parent, the node builds its own address by appending the parent address with its own address. Tree Cast differs from other approaches in a way that in this scheme addresses not only identify a node uniquely in the network, but also supports data routing depending on addresses. Routing based on assigned addresses is regarded as stateless routing since nodes do not preserve or maintain any state or routing history. when we have just one sink node, we can assign a null for its address and the nodes in the higher levels get the longer addresses. In that method, it is assumed that we are aware of the density of sensors within the simulation area and using this density we can set the number of bits in each level. But I think such assumption could lead to some unexpected errors in our addressing. **Because in that method we calculate density of sensors by dividing the number of node to the area of land but there is no information about the uniformity of the distribution of the nodes.** Here we describe this problem with a simple example:

Suppose we have an area of 1000 square meters and the number of nodes is 1000 sensors. So the density of nodes is 1 node per square meter. If the signal access zone of a sensor is 10 meters, the maximum number of nodes in the signal zone of sink node will be 10x10x3.14x1 (density) = 314 and we could use 9 bits for each level of addressing. Now imagine that we have the same density of nodes and the distribution of nodes is not uniform and the central area is much dense than the other parts. So it is possible that all the 1000 nodes are in the signal area of the sink node. We calculated 9 bits per level while here we cannot use 9 bits for addressing 1000 nodes connected to the sink node. In such situation using density number for calculating length of address cannot work and instead of density number we should use the number of nodes. In the example we described, having 1000 nodes, if we set 10 bits for addresses (10 bit is enough for maximum 1024 nodes) there will be no error in cases that there is no uniformity in the distribution. So using Tree Cast addressing method we should use the number of nodes not the density for making sure that we have enough size for addressing.

3. Unique ID assignment with Huffman algorithm

Our algorithm creates a 400x400 meters square as a virtual area of 16 hectares flat field. At the next step it produces a fixed node at the center of this square (x=200,y=200) that serves as the only sink node for our network. This program has designed to create a number of nodes ranging from 1 to 800. Also created nodes in random locations but we have implemented our work in it to distribute nodes with a uniform density within the square. After we created the nodes, one of the important steps of our work starts. The every single node finds a node in its neighborhoods that has the strongest signal and has a temporary address and if it finds such a node, sets that node as its parent. To find such a node, every node in the network broadcasts a signal to show its existence and then gathers signals from other nodes in its signal area. At first no node has an address but the sink node. Because of this at the first level of algorithm, the nodes neighboring to the sink node (in its signal access range) make a link with sink node and set the sink as their parent. In this part, sink node starts sending messages to each linked node and asks them to choose a temporary ID [18] for themselves. After receiving such message, connected nodes, randomly choose an ID and send it to sink node to confirm. If the ID is confirmed by the sink node, it sends a confirm message back to them and then the connected nodes append this chosen ID to the ID of their parent (the sink node) and make their own temporary ID. Then the sink node and the connected nodes update their parameters (e.g. its parent, its level, and its children). Sink node is at the level of zero and its children are in a level higher than it (Level one). To make our program more flexible, we have defined a variable that indicates the power of wireless signal of each node (the accessible radio range for nodes) and the program ask the user to set this variable. This variable is one of the key features of a wireless sensor and in the process of making a tree for our network, it has a big influence for the depth of the tree (the weaker the signal, the deeper the tree). We run this algorithm for every node in the network and every node finds the best parent for itself and at the end we have and interconnected balanced tree of linked nodes. The root of this tree is the sink node and a large number of the nodes are leaves of this tree. Due to the random location of nodes, if we set the signal power to a very low amount, it is possible that some nodes would be in an area with no nodes in signal access zone. At the end of this procedure such nodes remain unreachable and isolated from the network's tree. Besides in the cases that some nodes lose their battery and die, we can run this procedure and evaluate how losing a node (or nodes) could affect the accessibility and performance of the whole network. This tree of nodes is used to set temporary IDs for nodes. After assignment temporary ID, our algorithm to calculate the number of sub tree [18] for each node. After that we use this parameter to make a Huffman tree. Using Huffman codes we will reach out much less overhead for IDs. After using the sub tree number parameter of the sensor nodes and implements Huffman algorithm on this tree and makes Huffman codes for each node. These codes will be considered as the permanent address of each node in the network. When we generated Huffman codes, we can use the temporary IDs to reach out the deepest nodes and set its new permanent ID. And then set the permanent IDs for the nodes in a level less than those nodes and so on until the sink node.

```
void Huffman ()
{for( all the nodes in the network tree)
                calculate distance to the root;
        make (V_List); // list of nodes
        while (V_List is not empty)

        {sort ( V_List based on the distance of each node to the root);
                make a Virtual node;
                find ( two smallest nodes in the V_List);
                make( a HuffTree with Virtual node in the root and two smallest nodes left and right child of it);
                put the sum of two children in the Virtual node;
                remove two smallest nodes and add Virtual node to the V_List;
                sort (V_List);} // end of while
        for ( all the edges in the HuffTree)
        {label left edges with 0;
                label right edges with 1;}
        for (all the leaves in the HuffTree)
                traverse from root to the leaves and write the label of edges in the path;  }
```

```
void Huffman ()
{for( all the nodes in the network tree)
                calculate distance to the root;
        make (V_List); // list of nodes
        while (V_List is not empty)
        {
                sort ( V_List based on the distance of each node to the root);
                make a Virtual node;
                find ( two smallest nodes in the V_List);
                make( a HuffTree with Virtual node in the root and two smallest nodes left and right child of it);
                put the sum of two children in the Virtual node;
                remove two smallest nodes and add Virtual node to the V_List;
                sort (V_List);
        } // end of while
        for ( all the edges in the HuffTree)
        {
                label left edges with 0;
                label right edges with 1;
        }
        for (all the leaves in the HuffTree)
                traverse from root to the leaves and write the label of edges in the path;  }
```

4. Performance Evaluation

At this section we will compare our results (that we have got from Huffman code addressing) with the results that we get from the Tree Cast method. But here we show a simple network of sensors to make our descriptions more clear. In these figures we show how efficient the algorithm of Huffman coding is in comparison to the Tree Cast method. In this example we have a tree of 10 nodes. Using Tree Cast method we will need 120 bits of addresses in whole tree. If each one of the sensor nodes sends a data message to the root, we will have an average of 12 bits overhead (as address bits) for each message. Figure 1,2,3 and 4 shows methods steps.

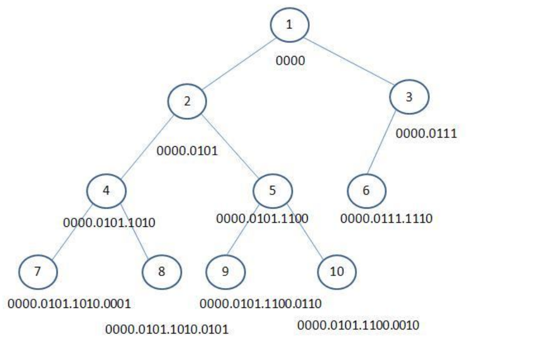

Figure 1 (using Tree Cast method for addressing)

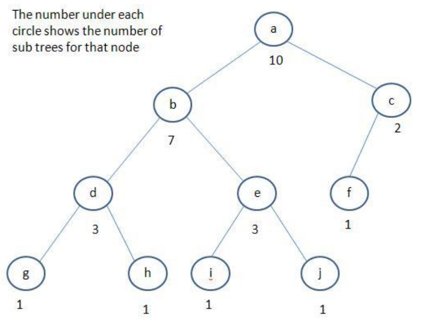
The number under each circle shows the number of sub trees for that node
a
10
b
7
c
2
d
3
e
3
f
1
g
1
h
1
i
1
j
1

Figure 2

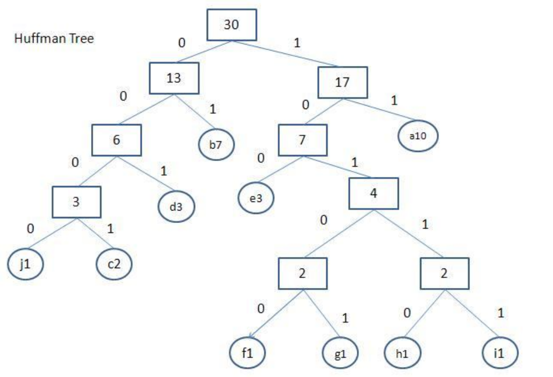

Figure 3

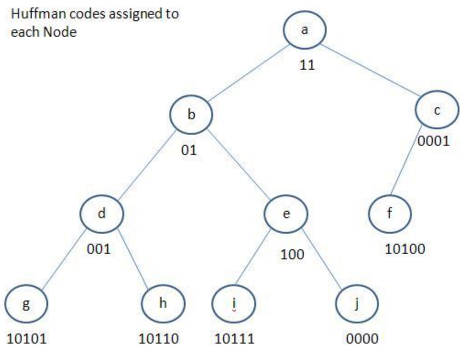

Figure 4

Using Huffman codes we will reach out much less overhead for addresses. Figure 4 shows that for our sample network, we need only 38 bits of address for whole network and if all the nodes send a data message to the root, we will have an average of 3.8 bits overhead (as address bits) for each message. That is dramatically less than the Tree Cast method.

5. SIMULATION RESULTS

We ran this algorithm multiple times with different values of signal power and with different number of nodes and we calculated the results for the Tree Cast method and the method we have developed and the results were amazing. The results of the program are the average number of bits needed for addressing a network of sensors. Here are our results:

Table 1 shows the results for networks that the number of nodes is in the range of 690 to 750.

Number of runs	Number of nodes	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Average Density of nodes	Depth of the network tree	Signal Zone (meters)
	705	11.32	19.24	38.48	0.004	7	50
	711	11.38	19.81	39.63	0.004	7	50
5	719	11.37	19.4	38.8	0.004	7	50
	729	11.37	19.13	38.26	0.004	7	50
	734	11.34	19.34	38.68	0.004	7	50
Average	719.6	11.356	19.384	38.77			
	706	10.89	16.9	28.18	0.004	5	70
	709	10.79	16.98	28.3	0.004	5	70
5	710	10.71	16.79	27.99	0.004	5	70
	711	10.79	16.91	28.18	0.004	5	70
	730	11	17.1	28.5	0.004	5	70
Average	713.2	10.836	16.936	28.23			
	697	10.67	16.08	22.97	0.004	4	90
	700	10.67	16.03	22.9	0.004	4	90
5	710	10.7	15.71	22.45	0.004	4	90
	715	10.7	15.82	22.6	0.004	4	90
	717	10.69	15.66	22.29	0.004	4	90
Average	707.8	10.686	15.86	22.642			
	697	10.72	15.16	18.95	0.004	3	110
	703	10.72	15.27	1909	0.004	3	110
5	716	10.73	15.05	18.81	0.004	3	110
	719	10.72	15.22..	19.03	0.004	3	110
	727	10.75	15.27	19.09	0.004	3	110
Average	712.4	10.728	15.194	18.994			
	708	10.65	13.92	17.4	0.004	3	130
	713	10.88	13.36	16.7	0.004	3	130
5	720	10.93	13.51	16.89	0.004	3	130
	725	10.87	13.34	16.67	0.004	3	130
	738	10.95	13.52	16.91	0.004	3	130
Average	720.8	10.856	13.53	16.914			
	708	10.65	13.92	17.4	0.004	3	130
	713	10.88	13.36	16.7	0.004	3	130
5	720	10.93	13.51	16.89	0.004	3	130
	725	10.87	13.34	16.67	0.004	3	130
	738	10.95	13.52	16.91	0.004	3	130
Average	720.8	10.856	13.53	16.914			
	683	10.81	12.68	14.08	0.004	2	170
	684	10.81	12.65	14.08	0.004	2	170
5	707	10.82	12.79	14.21	0.004	2	170
	714	10.82	12.95	14.38	0.004	2	170
	736	10.83	13.05	14.5	0.004	2	170
Average	704.8	10.818	12.824	14.25			

Table 2 shows the results for networks that the number of nodes is in the range of 530 to 590.

Number of runs	Number of nodes	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Average Density of nodes	Depth of the network tree	Signal Zone (meters)
	533	10.65	19.68	39.36	0.003	7	50
	547	11	20.15	40.29	0.003	7	50
5	558	10.8	19.75	39.5	0.003	7	50
	559	10.93	19.81	39.62	0.003	7	50
	564	10.87	19.18	38.37	0.003	7	50
Average	552.2	10.85	19.714	39.428			
	539	10.51	16.94	28.24	0.003	5	70
	545	10.56	19.92	28.2	0.003	5	70
5	549	10.54	17.19	28.65	0.003	5	70
	549	10.53	17.11	28.52	0.003	5	70
	558	10.54	17.35	28.92	0.003	5	70
Average	548	10.536	17.702	28.506			
	543	10.56	16.27	23.24	0.003	4	90
	544	10.56	16.08	22.98	0.003	4	90
5	547	10.61	15.87	22.67	0.003	4	90
	555	10.58	15.75	22.5	0.003	4	90
	568	10.6	15.96	22.8	0.003	4	90
Average	551.4	10.582	15.986	22.838			
	541	10.08	13.22	18.89	0.003	3	110
	549	10.23	13.18	18.83	0.003	3	110
5	557	10.26	13.37	19.1	0.003	3	110
	557	10.22	13.37	19.1	0.003	3	110
	558	10.21	13.27	18.96	0.003	3	110
Average	552.4	10.2	13.282	18.976			
5	538	10.76	13.62	17.03	0.003	3	130

	551	10.76	13.65	17.06	0.003	3	130
	552	10.76	13.38	16.72	0.003	3	130
	563	10.75	13.48	16.76	0.003	3	130
	565	10.74	13.41	16.76	0.003	3	130
Average	553.8	10.754	13.508	16.866			
	554	10.71	12.36	15.45	0.003	2	150
	561	10.7	12.41	15.51	0.003	2	150
5	568	10.76	12.37	15.46	0.003	2	150
	572	10.72	12.39	15.49	0.003	2	150
	573	10.77	12.5	15.62	0.003	2	150
Average	565.6	10.732	12.406	15.506			
	554	10.52	12.77	14.19	0.003	2	170
	555	10.52	12.68	14.09	0.003	2	170
5	557	10.56	12.85	14.27	0.003	2	170
	561	10.6	12.88	14.31	0.003	2	170
	566	10.55	12.96	14.4	0.003	2	170
Average	558.6	10.55	12.828	14.252			

Table 3 shows the results for networks that the number of nodes is in the range of 370 to 430.

Number of runs	Number of nodes	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Average Density of nodes	Depth of the network tree	Signal Zone (meters)
	395	10.26	16.46	41.14	0.002	7	50
	397	10.24	16.28	40.71	0.002	7	50
5	399	10.24	15.98	39.95	0.002	8	50
	401	10.26	15.81	39.53	0.002	7	50
	402	10.24	15.78	39.45	0.002	7	50
Average	398.8	10.248	16.062	40.156			
	386	10.1	14.15	28.29	0.002	5	70
	388	10.15	14.57	29.15	0.002	5	70
5	397	10.29	14.21	28.41	0.002	5	70
	404	10.25	14.84	29.68	0.002	5	70
	412	10.34	15.02	30.04	0.002	5	70
Average	397.4	10.226	14.558	29.114			
	390	9.66	13.58	22.64	0.002	4	90
	391	9.63	13.17	21.94	0.002	4	90
5	395	9.64	14.19	23.65	0.002	4	90
	401	9.65	13.63	22.72	0.002	4	90
	407	9.65	13.46	22.43	0.002	4	90
Average	396.8	9.646	13.606	22.676			
	388	9.65	13.31	19.02	0.002	3	110
	388	9.7	13.33	19.05	0.002	3	110
5	390	9.68	13.35	19.08	0.002	3	110
	391	9.7	13.43	19.18	0.002	3	110
	401	9.71	13.34	19.05	0.002	3	110
Average	391.6	9.688	13.352	19.076			
	384	9.96	11.65	16.64	0.002	3	130
	385	10.06	11.64	16.62	0.002	3	130
5	388	9.72	11.76	16.8	0.002	3	130
	393	9.73	11.97	17.1	0.002	3	130
	405	9.7	11.86	16.94	0.002	3	130
Average	391	9.834	11.776	16.82			
	370	9.81	12.32	15.41	0.002	2	150
	385	9.89	12.55	15.69	0.002	2	150
5	394	9.89	12.39	15.48	0.002	2	150
	399	9.96	12.41	15.51	0.002	2	150
	407	9.95	12.17	15.21	0.002	2	150
Average	391	9.9	12.368	15.46			
	379	9.8	11.57	14.46	0.002	2	170
	382	9.82	11.52	14.4	0.002	2	170
5	398	9.81	11.62	14.52	0.002	2	170
	404	9.82	11.41	14.26	0.002	2	170
	407	9.85	11.38	14.23	0.002	2	170
Average	394	9.82	11.5	14.374			

Table 4 shows the results for networks that the number of nodes is in the range of 210 to 270.

Number of runs	Number of nodes	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Average Density of nodes	Depth of the network tree	Signal Zone (meters)
	217	9.05	12.92	43.06	0.001	8	50
	221	9.01	13.05	43.48	0.001	8	50
5	236	9.13	13.23	44.11	0.001	8	50
	239	9.11	13.13	43.77	0.001	8	50
	243	9.12	13.19	43.05	0.001	8	50
Average	231.2	9.084	13.104	43.494			
5	227	9.29	11.81	29.52	0.001	5	70

	234	9.25	12.05	30.13	0.001	5	70
	237	9.3	11.34	28.35	0.001	5	70
	240	9.31	11.73	29.33	0.001	5	70
	248	9.33	11.69	29.23	0.001	5	70
Average	237.2	9.296	11.724	29.312			
5	219	8.64	11.42	22.83	0.001	4	90
	228	8.74	11.47	22.94	0.001	4	90
	232	8.87	11.4	22.8	0.001	4	90
	245	9.15	11.57	23.14	0.001	4	90
Average	229.4	8.812	11.396	22.79			
5	223	8.68	11.3	18.83	0.001	3	110
	224	8.68	11.52	19.2	0.001	3	110
	233	8.73	12.05	20.09	0.001	3	110
	235	8.71	11.39	18.98	0.001	3	110
	236	8.67	11.4	19.32	0.001	3	110
Average	230.2	8.694	11.532	19.284			
5	239	8.72	9.89	16.49	0.001	3	130
	240	8.72	10.13	16.88	0.001	3	130
	240	8.74	10.33	17.21	0.001	3	130
	241	8.73	10.38	17.3	0.001	3	130
	265	9.7	10.19	16.98	0.001	3	130
Average	245	8.922	10.184	16.972			
5	233	9.25	10.79	15.41	0.001	2	150
	235	9.31	10.81	15.45	0.001	2	150
	242	9.4	10.85	15.5	0.001	2	150
	242	8.74	11.51	16.45	0.001	2	150
	244	9.32	10.59	15.12	0.001	2	150
Average	239.2	9.204	10.91	15.586			
5	217	8.98	10.13	14.47	0.001	2	170
	220	8.89	9.61	13.73	0.001	2	170
	232	9.02	9.78	13.97	0.001	2	170
	237	9.15	9.95	14.22	0.001	2	170
	241	9.2	10.2	14.56	0.001	2	170
Average	229.4	9.048	9.934	14.19			

Table 5 and figure 5 show the average results based on the signal power of sensors for a number of sensors ranging from 690-750.

690-750				
The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree	Signal Zone (meters)
10.818	12.824	14.25	2	170
10.84	13.93	15.476	2	150
10.856	13.53	16.914	3	130
10.728	15.194	18.994	3	110
10.686	15.86	22.642	4	90
10.836	16.936	28.23	5	70
11.356	19.384	38.77	7	50

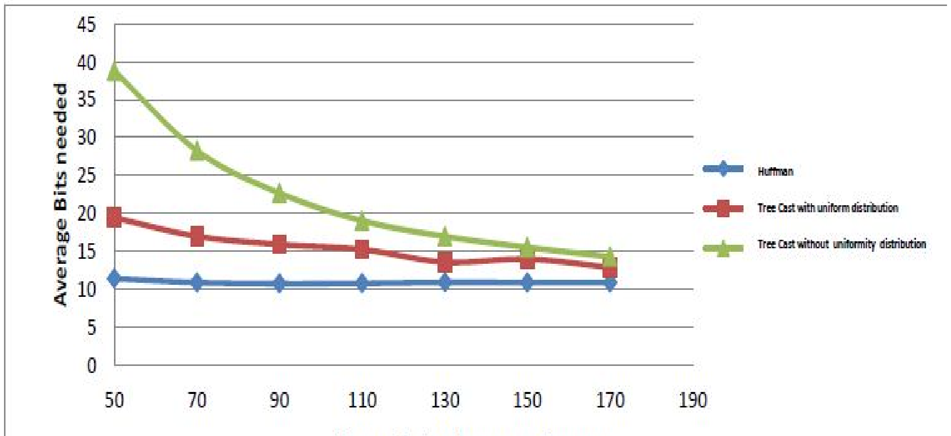

Figure 5.signal power of sensors

Table 6 and figure 6 show the average results based on the signal power of sensors for a number of sensors ranging from 530-590.

530-590				
The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree	Signal Zone (meters)
10.55	12.828	14.252	2	170
10.732	12.406	15.506	2	150
10.754	13.508	16.866	3	130
10.2	13.282	18.976	3	110
10.582	15.986	22.838	4	90
10.536	17.702	28.506	5	70
10.85	19.714	39.428	7	50

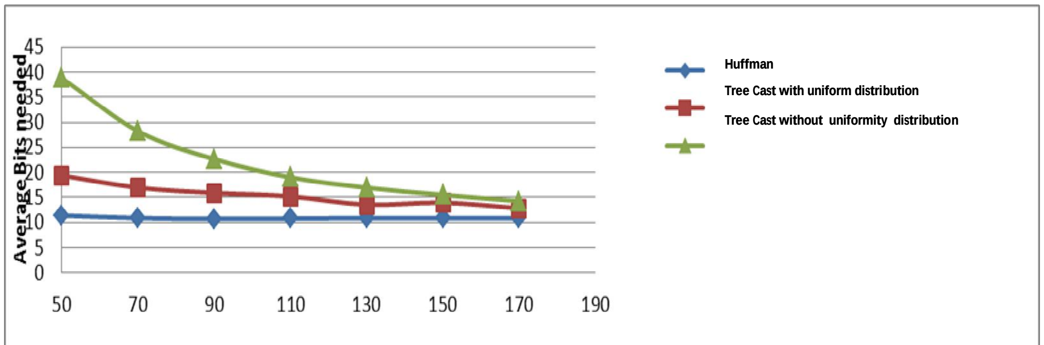

Figure 6.signal power of sensors

Table 7 and figure 7 show the average results based on the signal power of sensors for a number of sensors ranging from 370-430.

370-430				
The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree	Signal Zone (meters)
9.82	11.5	14.374	2	170
9.9	12.368	15.46	2	150
9.834	11.776	16.82	3	130
9.688	13.352	19.076	3	110
9.646	13.606	22.676	4	90
10.226	14.558	29.114	5	70
10.248	16.062	40.156	7	50

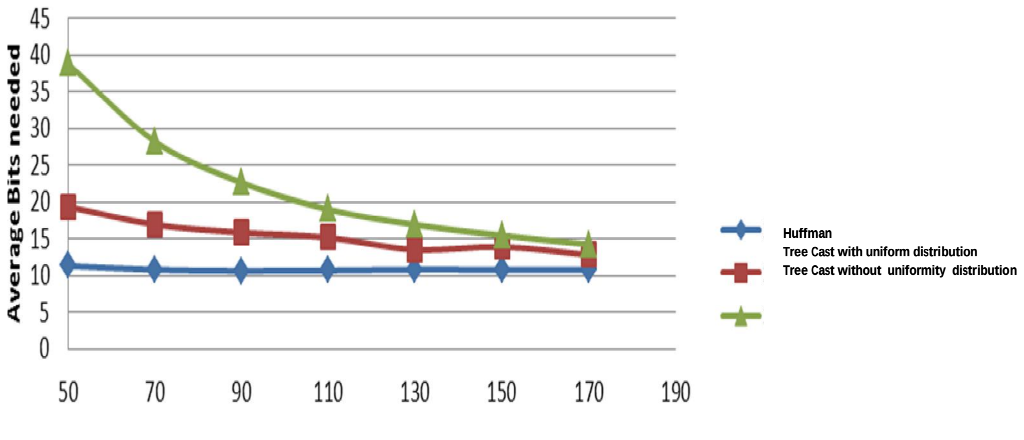

Figure 7.signal power of sensors

Table 8 and figure 8 show the average results based on the signal power of sensors for a number of sensors ranging from 210-270.

210-270				
The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree	Signal Zone (meters)
9.048	9.934	14.19	2	170
9.204	10.91	15.586	2	150
8.922	10.184	16.972	3	130
8.694	11.532	19.284	3	110
8.812	11.396	22.79	4	90
9.296	11.724	29.312	5	70
9.084	13.104	43.494	8	50

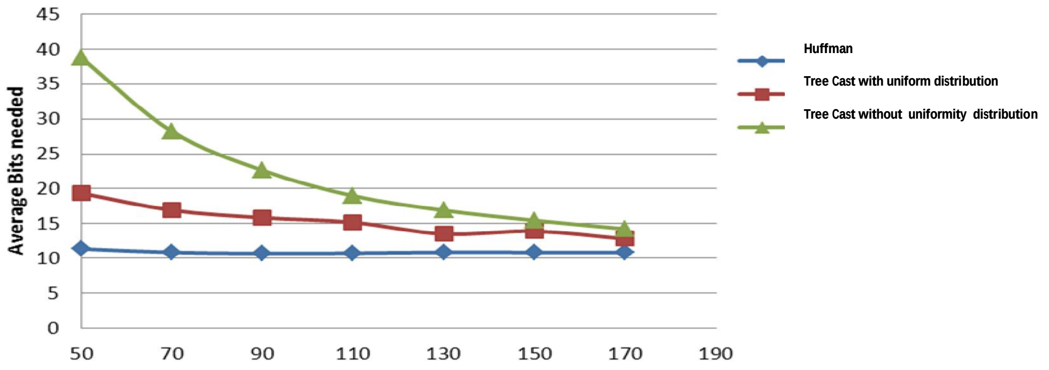

Figure 8.signal power of sensors

Table 9 to 15 and figures 9 to 15 show the effect of number of nodes on the length of bits we need for addressing using Tree Cast method and the method we have proposed while the power of signals are fixed to a certain value We assume there signal power of sensors is fixed .

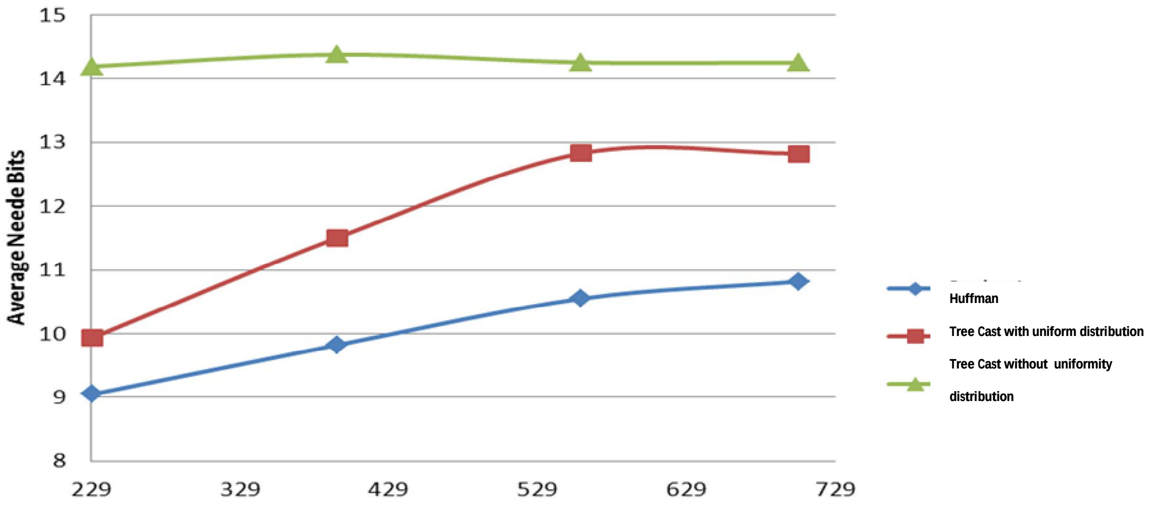

Figure 9. average number of nodes

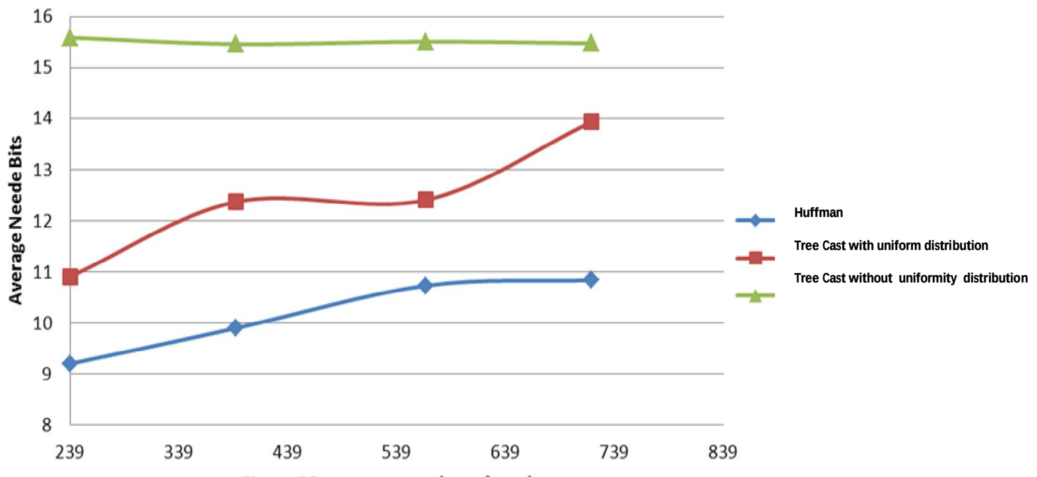

Figure 10. average number of nodes

The signal's access zone is 150 meters				
Average number of nodes we've tested	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree
718	10.84	13.93	15.476	2
565.6	10.732	12.406	15.506	2
391	9.9	12.368	15.46	2
239.2	9.204	10.91	15.586	2

The signal's access zone is 170 meters				
Average number of nodes we've tested	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree
704.8	10.818	12.824	14.25	2
558.6	10.55	12.828	14.252	2
394	9.82	11.5	14.374	2
229.4	9.048	9.934	14.19	2

The signal's access zone is 130 meters				
Average number of nodes we've tested	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree
720.8	10.856	13.53	16.914	3
553.8	10.754	13.508	16.866	3
391	9.834	11.776	16.82	3
245	8.922	10.184	16.972	3

The signal's access zone is 110 meters				
Average number of nodes we've tested	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree
712.4	10.728	15.194	18.994	3
552.4	10.2	13.282	18.976	3
391.6	9.688	13.352	19.076	3
230.2	8.694	11.532	19.284	3

The signal's access zone is 90 meters				
Average number of nodes we've tested	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree
707.8	10.686	15.86	22.642	4
551.4	10.582	15.986	22.838	4
396.8	9.646	13.606	22.676	4
229.4	8.812	11.396	22.79	4

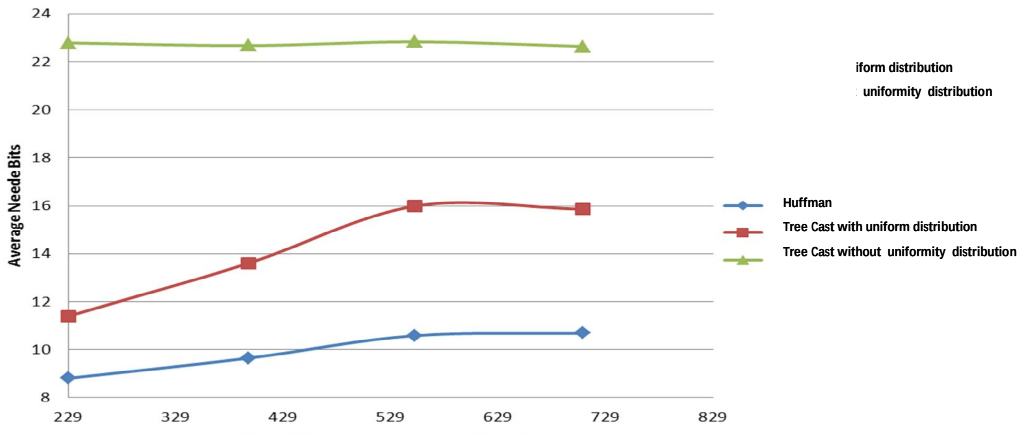

Figure 13. average number of nodes

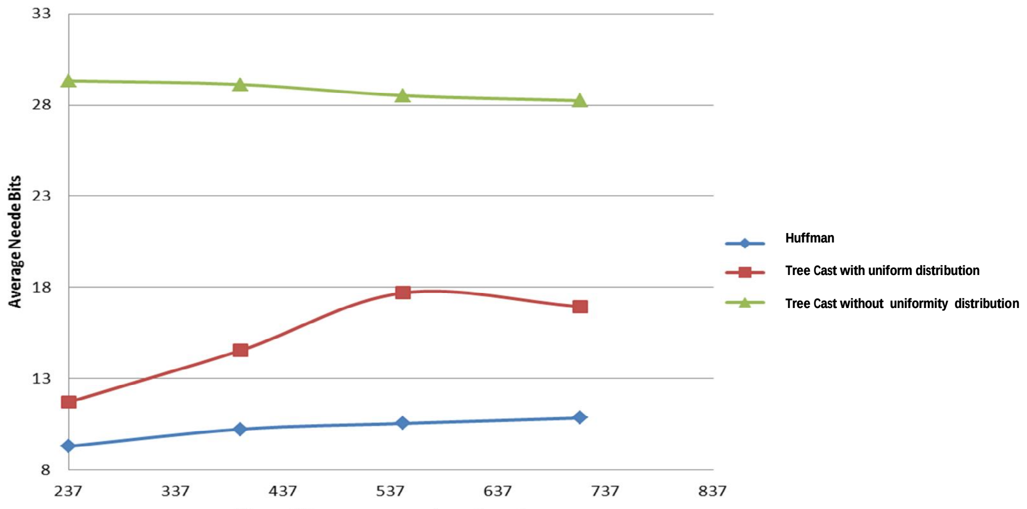

Figure 14. average number of nodes

The signal's access zone is 70 meters				
Average number of nodes we've tested	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree
713.2	10.836	16.936	28.23	5
548	10.536	17.702	28.506	5
397.4	10.226	14.558	29.114	5
237.2	9.296	11.724	29.312	5

The signal's access zone is 50 meters				
Average number of nodes we've tested	The average needed Bits using Huffman algorithm	The average Bits needed using Tree Cast method and with uniform distribution	The average Bits needed using Tree Cast method and without uniformity in distribution	Depth of the network tree
719.6	11.356	19.384	38.77	7
552.2	10.85	19.714	39.428	7
398.8	10.248	16.062	40.156	7
231.2	9.084	13.104	43.494	8

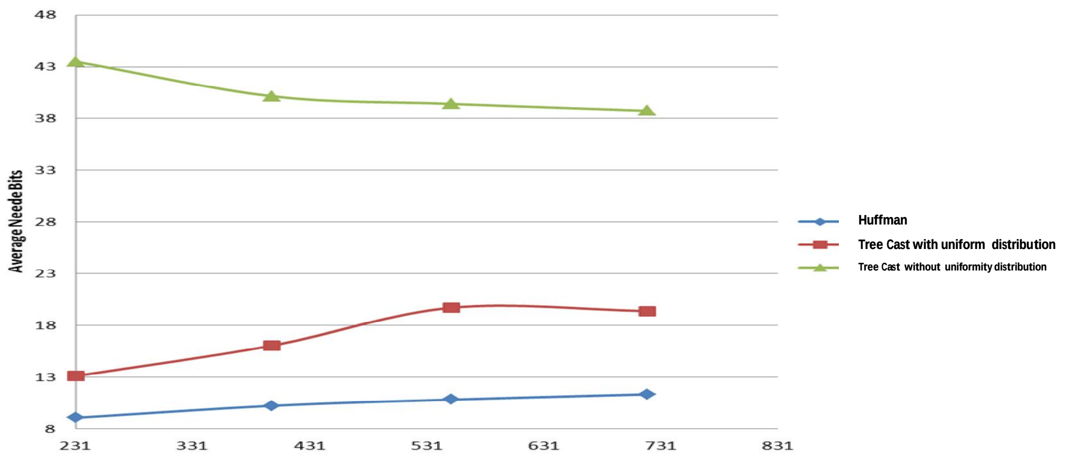

Figure 15. average number of nodes

As the diagrams show, the weak signal makes the network's tree deeper and the result is more bits needed for addressing in the Tree Cast method but the depth of tree doesn't have a big influence on the length of addresses in the Huffman method.

7. Conclusion

We presented a solution global ID assignment problem in sensor networks. Our solution of using Huffman code to assign unique identifiers for the nodes of a sensor network can save energy used by the sensors, In the first organizing nodes in a tree structure. This tree structure is used to compute the size of the network. Then we make a Huffman tree for unique Ids are assigned using the minimum number of bites. And evaluate its performance and compare results with tree cast method.

REFERENCES

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust commu- nication paradigm for sensor networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, pp. 56–67, August 2000.

[2] D. E. Y. Yu, R. Govindan, "Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks," Tech. Rep. TR-01-0023, UCLA/CSD, 2001.

[3] A. Dunkels, J. Alonso, and T. Voight, "Making tcp/ip viable for wireless sensor networks," in *European Workshop on Wireless Sensor Networks (EWSN)*, 2004.

[4] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermea-sures," in *IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[5] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor net- works," in *INFOCOM*, 2002.

[6] A. S. Tanenbaum, *Computer Networks*. Englewood Cliffs, 1989.

[7] J. R. Smith, "Distributing identity," *IEEE Robotics and Automation Magazine, Vol.6, No.1*, March 1999.

[8] C. Schurgers, G. Kulkarni, and M. B. Srivastava, "Distributed on-demand address assignment in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems, Vol.13, No.10*, October 2002.

[9] D. Estrin, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *MOBICOM*, 1999.

[10] M. Ali and Z. A. Uzmi, “An energy efficient node address naming scheme for wireless sensor networks,” in *INCC*, 2004.

[11] H. Zhou, M. W. Mutka, and L. M. Ni, “Reactive id assignment for wireless sensor networks,” *International Journal of Wireless Information Networks, Vol.13, No.4*, October 2006

[12] M. Ali and Z. A. Uzmi, .An energy ef_cient node address naming scheme for wireless sensor networks,. in *INCC*,2004

[13] S. Motegi, K. Yoshihara, and H. Horiuchi, .Implementation and evaluation of on-demand address allocation for event-driven sensor network,. in *Proceeding of the Symposium on Applications and the Internet (SAINT’05)*, 2005.

[14] F. Kuhn, T. Moscibroda, and R. Wattenhofer, .Initializing newly deployed ad hoc and sensor networks,. In *Proceedings of the 10th annual international conference on Mobile computing and networking (MobiCom’04)*, 2004.

[15] M. Kochhal, L. Schwiebert, and S. Gupta, .Role-based hierarchical self organization for wireless ad hoc sensor networks,. in *Proceedings of the Second ACM International Workshop on Wireless Sensor Networks and Applications (WSNA’03)*, 2003.

[16] K. Sohrabi, V. Ailawadhi, J. Gao, and G. Pottie, .Protocols for self organization of a wireless sensor network,. *Personal Communication Magazine*, vol. 7, 2000.

[17] S. PalChaudhuri, S. Du, A. K. Saha and D. B. Johnson, "TreeCast: A Stateless Addressing and Routing Architecture for Sensor Networks", Proc. of the 18th IPDPS International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN 2004), p. 221a, Santa Fe, New Mexico, April 2004.

[18] E. Ould-Ahmed-Vall, D. M. Blough, B. S. Heck and G. F. Riley, "Distributed Global Identification for Sensor Networks", Technical Report 2005, Georgia Tech, 2005.