TURKISH NAVAL ACADEMY
NAVAL SCIENCE AND ENGINEERING INSTITUTE
DEPARTMENT OF COMPUTER ENGINEERING

# THE DISTRIBUTED 3D SPACE COVERAGE SCHEMES FOR UNDERWATER WIRELESS SENSOR NETWORKS

MASTER THESIS

**HAKAN TEZCAN**

150524

Advisor : Assoc.Prof. Erdal Çayırcı
Co-Advisor : Assist.Prof. Vedat Coşkun

İstanbul, 2004

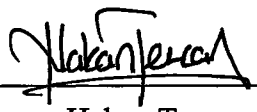
**THE DISTRIBUTED 3D SPACE COVERAGE SCHEMES FOR UNDERWATER WIRELESS SENSOR NETWORKS**

Submitted in partial fulfillment of the requirements for degree of

**MASTER OF SCIENCE IN COMPUTER ENGINEERING**

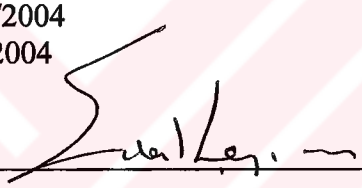
from the

**TURKISH NAVAL ACADEMY**

Author : 

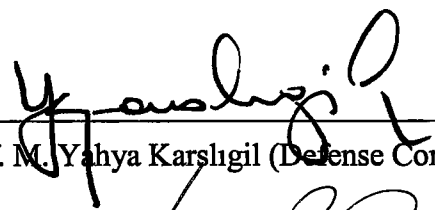
Hakan Tezcan

Defense Date : 26/07/2004
Approved by : 26/07/2004

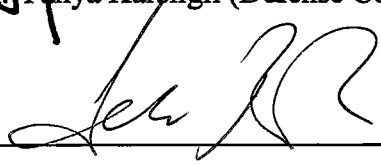


Assoc.Prof. Erdal Çayırcı (Advisor)

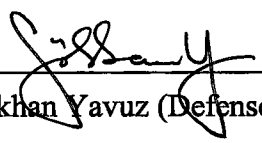
Assist. Prof. Vedat Coşkun (Co-Advisor)



Prof. M. Yahya Karslıgil (Defense Committee Member)



Prof. Şebnem Baydere (Defense Committee Member)



Assist.Prof. Gökhan Yavuz (Defense Committee Member)

## ABSTRACT (TURKISH)

# SUALTI TELSİZ DUYARGA AĞLARI İÇİN DAĞITIK ÜÇ BOYUTLU UZAY KAPSAMA YAPILARI

*Anahtar Kelimeler :* Telsiz duyarga ağlar, Sualtı telsiz duyarga ağlar, Kapsama, Yerleştirme, Dağıtık algoritma, Taktiksel gözetleme.

Mikro Elektro-Mekanik Sistemlerde (MEMS) ve telsiz ağ teknolojilerinde son zamanlarda meydana gelen bilimsel gelişmeler her zaman ve her yerde bilgiye erişim sağlayan duyarga adı verilen küçük telsiz cihazlarla bizleri yeni bir çağın eşiğine getirmiştir. Geleneksel bilgisayar ağlarının aksine telsiz duyarga ağları çok fazla sayıdaki bu çok küçük otonom duyargalardan oluşur. Her bir duyarga güç kaynağı olarak pil kullanır ve entegre sensörler, bilgi işleme ve kısa mesafe iletişim yeteneği ile donatılmıştır. Duyarga ağda görevli tüm ağ elemanları çevrelerinden veri toplayarak bu verileri bir işbirliği içerisinde düşük güç tüketimiyle herhangi bir son noktaya iletirler.

Bu tezde menzilindeki bir hedefi algılamak amacıyla kullanılabilecek Sualtı Telsiz Duyarga Ağları için yeni bir mimari öneriyoruz. Bizim mimarimize göre, duyargalar su üstüne rastsal olarak dağıtıldıklarında şamandıraların içinde bulunurlar. Duyargaların dağıtılmasından sonra, aynı zamanda şamandıra ve duyarga arasında iletişim hattı olarak da kullanılan bir kablo vasıtasıyla duyargalar kendi derinliklerini ayarlarlar. Duyargalar sualtında olmasına rağmen şamandıralarda bulunan antenler yoluyla su üstündeki telsiz ortamı kullanarak birbirleriyle işbirliği halinde çalışırlar.

Duyarga ağlarında kapsama bir alanın ne kadar iyi izlendiğini gösterir ve ağın servis kalitesinin bir ölçütü olarak düşünülebilir. Duyarga alanındaki her bir noktanın en az bir duyarganın algılama menzili içinde bulunması iyi bir servis kalitesi için arzu edilir. Bu nedenle, aynı zamanda önerdiğimiz mimarinin üç boyutlu duyarga uzayının kapsamasını azami arttırmak amacıyla iki dağıtık

yerleştirme algoritması geliştirdik. Bu dağıtık yerleştirme algoritmalarının ilki mevki bilgisine sahip duyargaların oluşturduğu duyarga ağları diğeri için diğeri ise mevki bilgisinden yoksun duyargalardan oluşan duyarga ağları için geliştirildi. Yalnızca yerel paket alışverişine dayanan dağıtık yerleştirme algoritmalarımız duyargalar üzerinde çalışır ve duyarga uzayının üç boyutlu kapsamasını azami arttıracak şekilde komşu duyargaların mevkilerine göre duyargaların derinliğini hesaplar. Geliştirdiğimiz yapılar dağıtık ve uyarlanabilir olmasına rağmen kabul edilebilir bir kontrol trafik yükü ile duyarga uzayında yüksek bir kapsama sağlar.

## ABSTRACT (ENGLISH)


### THE DISTRIBUTED 3D SPACE COVERAGE SCHEMES FOR UNDERWATER WIRELESS SENSOR NETWORKS

*Keywords:* Wireless sensor networks, Underwater wireless sensor networks, Coverage, Placement, Distributed algorithm, Tactical surveillance.

Recent scientific advances in Micro Electro-Mechanical Systems (MEMS) and wireless network technologies have placed us at the doorstep of a new era where small wireless devices which are called sensor nodes will provide access to information anytime, anywhere. Contrary to more traditional computer networks, wireless sensor networks consist of a large number of these ultra-small autonomous sensor nodes. Each sensor node is battery powered and equipped with integrated sensors, data processing capabilities, and short range radio communications. In wireless sensor networks, sensor nodes gather data from their environment and convey these data to any end point using a wireless medium with a collaborated effort by using their limited power sources.

In this thesis, we propose a novel architecture for underwater wireless sensor networks that can be used to detect a target in the vicinity of the sensor nodes. According to our architecture, when sensor nodes first deployed randomly, sensors lie in surface buoys. After deployment, nodes adjust their depths via a cable which is also used for the communication link between the sensor and the surface buoy. Although the sensors are underwater, the nodes can collaborate through the wireless medium over sea surface by using the antenna at the surface buoys.

In wireless sensor networks, coverage represents how well an area is monitored and can be considered as the measure of Quality of Service (QoS) of

the network. It is most desirable that each point in the sensor field should be within the sensing range of at least one sensor for achieving a good QoS. Therefore we also develop two distributed placement algorithms in order to maximize the coverage of the 3D underwater sensor space of our proposed architecture. One of these algorithms is developed for sensor networks consist of location aware nodes and the other one is for the sensor networks consist of non-location aware nodes. Our distributed placement algorithms that rely solely on local packet exchanges run on sensor nodes and calculate the depths of sensors according to the locations of neighboring nodes such that the maximum three dimensional coverage of the sensor space is maintained. Although our schemes are distributed and adaptive, they maintain a high coverage of the sensor space in the expense of acceptable control traffic overhead.

## DISCLAIMER STATEMENT

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Turkish Navy, Naval Academy, and Naval Science and Engineering Institute.

## DEDICATION

*In memory of my father, mother, and sister.*

## ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Assoc.Prof. Erdal Çayırcı for his intelligent supervision, inspiration and motivating suggestions. I also would like to thank Assist.Prof. Vedat Coşkun for his helpful discussions and Prof. Süleyman Özkaynak for obtaining the required materials and support.

Finally, I would like to express my deepest gratitude to my brother Ceyhan Tezcan for his love, patience and everlasting support.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

## LIST OF ABBREVIATIONS, ACRONYMS, AND SYMBOLS

| | |
|---|---|
| **3D** | Three Dimensional |
| **2D** | Two Dimensional |
| **QoS** | Quality of Service |
| **WSNs** | Wireless Sensor Networks |
| **MEMS** | Micro Electro-Mechanical Systems |
| **NBC** | Nuclear, Biological and Chemical |
| **SDVs** | Small Delivery Vehicles |
| **ILP** | Integer Linear Programming |
| **NP** | Nondeterministic Polynomial-Time |
| **AGP** | Art Gallery Problem |
| **VFA** | Virtual Force Algorithm |

# I. INTRODUCTION

## A. WIRELESS SENSOR NETWORKS

Recent scientific advances in Micro Electro-Mechanical Systems (MEMS) and wireless network technologies have placed us at the doorstep of a new era where small wireless devices which are called sensor nodes will provide access to information anytime, anywhere [1][2]. Contrary to more traditional computer networks, wireless sensor networks (WSNs) consist of a large number of these ultra-small autonomous sensor nodes. Each sensor node is battery powered and equipped with integrated sensors, data processing capabilities, and short range radio communications [3][4]. The basic components of a typical sensor node are illustrated in Figure 1 [1].

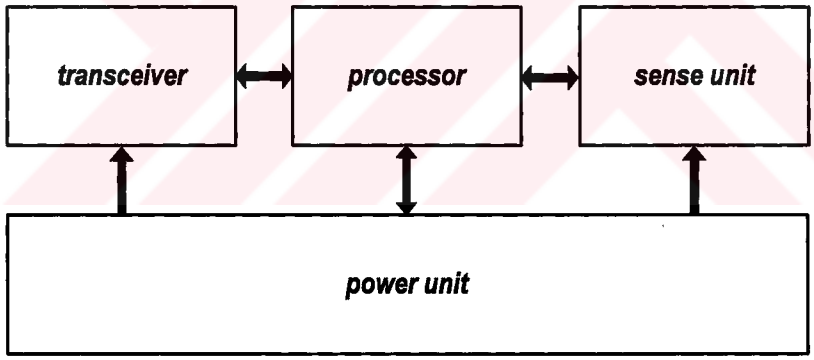



*Figure 1. A typical sensor node units [1].*

In WSNs, we usually confront the following three terminologies [2]:

- *Sensor node:* The device that implements the physical sensing of environmental phenomena and reporting of measurements through wireless communication.

- *Observer:* The end user interested in obtaining information disseminated by the WSNs about the phenomenon. The observer may indicate interests or queries to the network and receive responses to these queries.

- *Phenomenon*: The entity of interest to the observer that is being sensed and optionally analyzed/filtered by the WSNs. There may be multiple phenomena under observation concurrently in the same network.

To maintain sufficient coverage and fault-tolerance a large number of sensor nodes ranging from hundreds to several thousands [5][6] are deployed to form a WSN. Sensor nodes usually are thrown in mass or placed one by one either inside or very close to a phenomenon. Networking these sensor nodes which coordinate amongst themselves to govern a global sensing task will make major changes in information gathering both in remote inhospitable area and urban environments because of their reliability, accuracy, flexibility, cost-effectiveness and easy of deployment [5].

WSNs may consist of many different types of sensors such as seismic, low sampling rate magnetic, thermal, visual, infrared, acoustic, and radar, which are able to monitor a wide variety of ambient conditions that include the following [1][5]:

- temperature,
- humidity,
- vehicular movement,
- lightning condition,
- pressure,
- soil makeup,
- noise levels,
- the presence or absence of certain kinds of objects,
- mechanical stress levels on attached objects, and

- the current characteristics such as speed, direction, and size of an object.

The concept of micro sensing and wireless connection of sensor nodes promise many application areas. In [1], application areas of WSNs are categorized as follows:

a. *Military Applications*
   - Monitoring friendly forces, equipment and ammunition
   - Battlefield surveillance
   - Reconnaissance of opposing forces and terrain
   - Targeting
   - Battle damage assessment
   - Nuclear, biological and chemical (NBC) attack detection, and reconnaissance

b. *Environmental Applications*
   - Forest fire detection
   - Biocomplexity mapping of the environment
   - Flood detection
   - Precision agriculture

c. *Health Applications*
   - Telemonitoring of human physiological data
   - Tracking and monitoring doctors and patients inside a hospital
   - Drug administration in hospitals

d. *Home Applications*
   - Home automation
   - Smart environments

e. *Space Exploration Applications*

f. *Chemical Processing Applications*

g. *Disaster Relief Applications*

*h. Other Commercial Applications*

- Environmental control in office buildings
- Interactive museums
- Detecting and monitoring car thefts
- Managing inventory control
- Vehicle tracking and detection

As we realize above applications we can clearly see that WSNs need wireless ad hoc networking techniques. Although many protocols and algorithms have been proposed for traditional wireless ad hoc networks, they are not viable options for WSNs because of the different characteristics of these networks such that sheer number of sensor nodes, dense deployment, high node failure rate, limited and unreplenishable on board energy, low computational and memory capacities and frequent topology changes [1][7].

The most fundamental constraint of WSNs is the energy bottleneck. Sensor nodes have limited, usually irreplaceable power sources. Therefore WSNs protocols must primarily focus on the power conservation different from traditional networks that promise to achieve high QoS [8]. They should give options to end user the ability to trade off between the energy efficiency/system lifetime, latency, accuracy, and fault-tolerance [1][2][9][10].

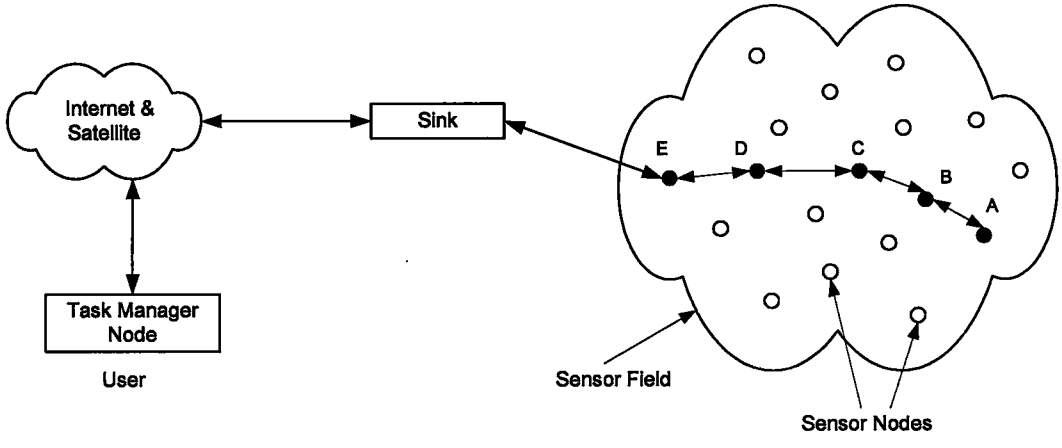



*Figure 2. A sample WSN [1].*

Communication in WSNs differs from communication in other types of networks in that it is not end to end [11]. In general, sensor nodes are scattered in a *sensor field* randomly. A sample sensor network is shown in Figure 2 [1]. Each of these sensor nodes can collect data and conveyed back to the *sink* and the end users. Data are conveyed to the end user by a multihop infrastructureless architecture through the *sink*. The *sink* may communicate with the task manager node via internet or Satellite.

## B. PROBLEM DEFINITION AND PROPOSED SCHEME

WSNs are based on the collaborative effort of vast number of tiny sensor nodes embedded in the phenomenon to be observed [1]. Each sensor node is battery powered and equipped with integrated sensors, data processing capabilities, and short range radio communications [3][4]. In WSNs, sensor nodes gather data from their environment and convey these data to any end point using a wireless medium with a collaborated effort by using their limited power sources.

WSNs will permit remote object monitoring and tracking in many different context [5]. One of the best application areas for WSNs is tactical underwater surveillance systems. We introduce a wireless underwater sensor network architecture where sensor nodes initially lie in the surface buoys. After the deployment of these buoys, sensor nodes can be lowered to various depths such that the maximum coverage of the 3D sensor space is maintained.

The salient features of this architecture can be summarized as follows:

- Each surface buoy has a bunch of sensors. These sensors can be lowered to a calculated depth via a cable, i.e. the sensors of the same node stay at the same depth. This is illustrated in Figure 3.

- The cable that connects sensors to the surface buoy is used for the communication between sensors and the buoy.

- By choosing the wired media as the communication link between sensors and the surface buoy, we eliminate the difficulties of acoustic media such as severe bandwidth limitation, high latency, multipath, signal fading, reverberation, and environmental noise [12]. Sensor nodes communicate with each other through the wireless medium over sea surface by using the antennae at the surface buoys.

- The buoys collect sensed data from their sensors and convey this data to the data collecting buoy (*cbuoy)*, i.e. the sink. Please note that we call a buoy and the sensors attached to it as a sensor node.

- Uncovered zones that can exist due to sensor node failures or mobility can be hindered by rearranging the depths of the sensor nodes.

- The *cbuoy* is the gateway between the network and the users of the system.

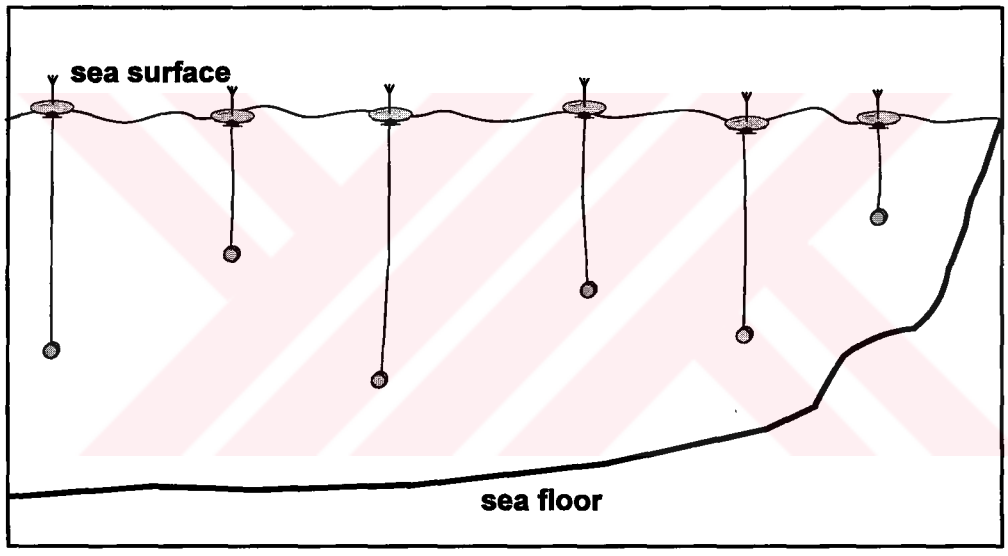



*Figure 3. Underwater wireless sensor network.*

Our architecture is for tactical underwater surveillance systems that can be used to detect enemy submarines, small delivery vehicles (SDVs), mines, and divers. We also introduce two distributed algorithms that fit the requirements of this application area. One of these algorithms is developed for sensor networks consist of location aware nodes and the other one is for the sensor networks consist of non-location aware nodes. Our algorithms run on sensor nodes and calculate the depths of sensors according to the locations of neighboring nodes such that the maximum three dimensional coverage of the sensor field is

maintained. Our algorithm is an adaptive one that can rearrange the depths of sensor nodes as they move by currents, winds or other reasons.

### C. CONTRIBUTION OF THIS THESIS

In this thesis we explore coverage and placement issues in WSNs. We introduce a novel architecture for underwater wireless sensor networks that can be used to detect enemy submarines, SDVs, mines and divers. We also propose two distributed algorithms that fit the requirements of this application area. After deployment, sensor nodes run our algorithms and calculate the depths of sensors according to the locations of neighboring nodes such that the maximum three dimensional coverage of the sensor space is maintained. Although our algorithms are distributed and adaptive, they maintain a high coverage of the sensor space in the expense of acceptable control traffic overhead.

### D. STRUCTURE OF THIS THESIS

This thesis is organized as follows:

In Section II we present background information about coverage and placement issues in WSNs, and discuss the related work. Our coordinate system model, the formal definition of the problem and constraints are explained and our distributed algorithms for 3D space coverage of sensor space are presented in Section III. The performance of the schemes is evaluated in Section IV.

Finally, Section V gives concluding remarks of this thesis.

## II. BACKGROUND INFORMATION AND RELATED WORK

### A. COVERAGE IN WIRELESS SENSOR NETWORKS

One of the fundamental issues that arises in WSNs, in addition to location calculation, tracking, and deployment, is coverage. Due to the large variety of sensors and applications, coverage is subject to a wide range of interpretations. In WSNs, coverage represents how well an area is monitored and can be considered as the measure of QoS of the network [13][14]. As we consider each sensor node typically has a physical sensing range within which it is able to perform its operation, the ultimate purpose of a sensor network is that that each point in the sensor field should be within the sensing range of at least one sensor for achieving a good QoS [15].

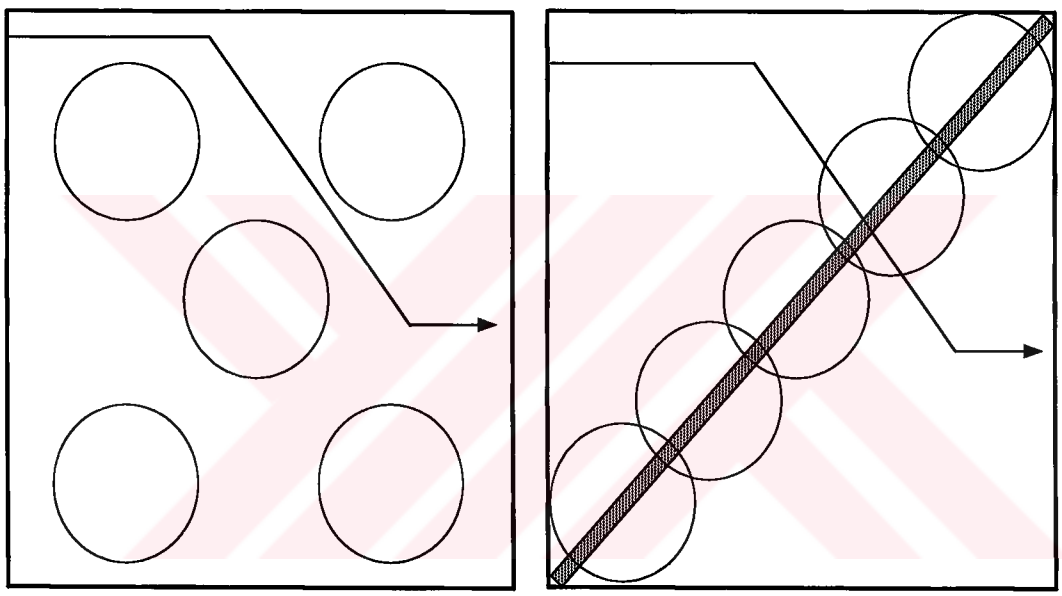
In [16] coverage behaviors are classified into three varieties:

1. ***Blanket (or area) coverage***: The objective of blanket coverage is to achieve a static arrangement of sensor nodes that maximizes the detection rate of targets appearing within the coverage area. Blanket coverage also known as area coverage [17] and grid coverage [18][19][20].

2. ***Barrier coverage***: The objective of barrier coverage is to achieve a static arrangement of sensor nodes that minimizes the probability of undetected penetration of a target through the barrier.

3. ***Sweep coverage***: The objective of sweep coverage is to move a number of sensor nodes across a coverage area in a manner which addresses a specified balance between maximizing the number of detections per time and minimizing the number of missed detections per area. Sweep coverage can be considered as a moving barrier consisting of mobile sensor nodes.

The difference between area coverage and barrier coverage is illustrated in Figure 4. The arrowed line represents the possible path of the intruder and the diagonal line shows the barrier. For the area coverage, sensor nodes are deployed in order to maximize covered area, but for that case, a traversing target can likely find an uncovered path without being recognized. For the case of barrier coverage, sensor nodes are deployed to detect a traversing target but they have less probability to detect a target that appear anywhere in the sensor field.



a. Area Coverage b. Barrier Coverage

*Figures 4. Difference between area and barrier coverage.*

### 1. Mathematics of Coverage Paradigm

Since WSNs are made up of a sheer number of sensor nodes, coverage of the network should be thought as the total coverage of all the sensor nodes in the network. For each individual sensor node, coverage paradigm can be modeled in two different ways based on the accepted sensor detection type [21][22][23].

### a. Coverage Model Based On Binary Sensor Detection

According to the binary sensor detection type, a simplifying assumption is made for coverage. Sensor readings have no associated uncertainty and a sensor node always detects (covers) a target that presents within its range.

Consider an $n \times n \times n$ sensor grid and assume that there are $k$ sensor nodes deployed. Each sensor node has a detection range $r$. Assume sensor node $s_i$ is deployed at point $(x_i, y_i, z_i)$. For any point $P$ at $(x, y, z)$, the Euclidean distance between $s_i$ and $P$ as $d(s_i, P)$ can be denoted as follows;

$$d(s_i, P) = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} \quad (1)$$

Equation 2 shows the binary sensor detection model that expresses the coverage $c_{xyz}(s_i)$ of a point $P$ by sensor $s_i$.

$$c_{xyz} = \begin{cases} 1, \textit{if } d(s_i, P) < r \\ 0, \textit{otherwise.} \end{cases} \quad (2)$$

### b. Coverage Model Based On Probabilistic Sensor Detection

In real life scenario, sensor detections are imprecise. Therefore coverage can be modeled with probabilistic terms also.

Consider a sensor field with $n$ sensor deployed. A target at location $u$ emits a signal which is measured by the sensor nodes. The signal from the target decays as a polynomial of the distance. If the decay coefficient is $k$, the signal energy of a target at location $u$ measured by the sensor at $s_i$ is given by

$$S_i(u) = \frac{K}{\|u - s_i\|^k} \tag{3}$$

where $K$ is the energy emitted by the target and $\|u - s_i\|^k$ is the Euclidean distance between the target and the sensor node. Depending on the environment the value of $k$ typically varies from 2 to 5.

Energy measurements at a sensor are usually corrupted by noise. If $N_i$ denotes the noise energy at sensor $s_i$ during a particular measurement, then the total energy measured at sensor $s_i$, when the target is at location $u$, is

$$E_i(u) = S_i(u) + N_i = \frac{K}{\|u - s_i\|^k} + N_i \tag{4}$$

The sensors collaborate to arrive at a consensus decision as to whether a target is present in the region. There are different approaches for reaching this consensus [24]. For example in value fusion, one of the sensor node gathers the energy measurements from the other sensors, totals up the energy and compares the sum to a threshold to decide whether a target is present. If the sum exceeds the threshold, then the consensus decision is that a target is present. The probability of consensus target detection by using value fusion i.e., $D_v(u)$ can be written as:

$$D_v(u) = prob\left[\sum_{i=1}^{n} \frac{K}{\|u - s_i\|^k} + N_i\right] \geq \eta \tag{5}$$

where $\eta$ is the value fusion threshold.

## B. NODE PLACEMENT IN WIRELESS SENSOR NETWORKS

Large numbers of inaccessible and unattended sensor nodes, which are prone to frequent failures, make topology maintenance a challenging task [1]. Hundreds to several thousands of nodes are deployed throughout the sensor field. In WSNs, deployment affects coverage, communication cost and resource management. Sensor nodes may be deployed by two different ways:

1. *Manually*
2. *Autonomously*

When sensor nodes deployed manually, nodes can be either thrown in mass or placed one by one in the sensor field. They can be deployed by

- dropping from a plane,
- delivering in an artillery shell, rocket or missile,
- throwing by a catapult (from a ship board, etc.),
- placing in factory, and
- placing one by one either by a human or a robot.

Autonomous deployment is used for mobile sensor nodes to move themselves to sensing locations from an initial arrangement as described above.

### 1. Deployment Strategy

Deployment issue has vital importance in WSNs. For successful operation of the network, the deployment should result in configurations that not only provide good "sensor coverage" but also satisfy certain local and global connectivity [25]. Since the wireless radio in each sensor node also has a transmission range, the placements of the nodes determine the coverage and connectivity of the WSN. As sensor nodes need to organize themselves into a

connected ad-hoc network, connectivity of the nodes should be maintained while maximizing the coverage of sensor field. Without sufficient coverage, the network cannot guarantee the quality of monitoring service. Without network connectivity, nodes may not be able to send data back to the remote end user [26]. Furthermore a good deployment strategy should not disregard redundancy due to the node failures because of the environmental physical conditions or energy deficiency of nodes.

We classify the related work about the deployment strategies in three categorizations:

**a. Random Deployment**

Random deployment is used in remote or inhospitable areas that have severe resource constraints. According to random deployment pattern sensor nodes are scattered into random locations. When random deployment is used the location of each sensor node cannot be fully guaranteed a priori [27]. Therefore, the number of sensor nodes that must be deployed in order to completely cover the whole monitored area is often higher than if a deterministic procedure were used.

A mechanism for sensor collaboration to perform target detection is proposed and analyzed to evaluate the "exposure" (target detection probability) of paths for barrier coverage in [22]. The focus of this approach is to determine the number of sensors to be deployed to carry out target detection in a region of interest. The strategy consists of sequentially deploying a limited number of sensors at a time into 2D sensor field until the desired minimum exposure is achieved.

Computational geometry and graph theoretic techniques, specifically the Voronoi diagrams and graph search algorithms are combined and a polynomial-

time algorithm is presented for coverage calculation in [13]. Proposed algorithm finds lowest coverage path which maximizes the distance of the path to all sensor nodes and highest coverage path which minimizes the distance of the path to closest sensor nodes. Additional sensor deployment heuristics are also given to improve stochastic coverage in [13].

Coverage was focused as the main criterion on power aware operation strategies for WSNs in [28]. By using integer linear programming (ILP) based formulations and strategies having a limited set of sensor nodes active at any given time is the key in reducing the overall energy consumption of the network since unused sensors can be placed in special power-saving sleep modes while maintaining guaranteed sensor coverage. The targeted problem of the authors is NP-complete and their approach does not appear scalable due to its computational complexity.

[29] is the first to propose a "differentiated surveillance" service for WSNs with minimal overhead. Authors argue that in most scenarios, there are certain geographic regions in the sensor field that are much more security-sensitive than others and need preferential coverage. Based on the fact that individual sensor nodes are not reliable, prone to failures and single sensor readings can be easily distorted by background noise; they develop a differentiated surveillance service which provides higher degree of coverage in which multiple sensor nodes monitors the same location in order to obtain high confidence level coverage for the critical regions in the sensor field. According to the sensing coverage algorithm, each node is able to dynamically decide a schedule for itself to guarantee a certain degree of coverage with average energy consumption inversely proportional to the node density. Main difference of this approach that provides full sensing coverage in the context of energy conservation is maintaining the degree of coverage up to the limit imposed by the density of sensor nodes.

In [14] asymptotic behaviors of the coverage and detectability of WSNs are characterized between randomly deployed WSNs and 2D grid-based WSNs. While grid-based WSNs provide efficient area coverage, random networks offer robustness and reliability upon sensor failures at the cost of coverage redundancy. Main goal of the study is deriving a node density value to achieve area coverage. According to the proposed work, in order to effectively detect any crossing object in the sensor field, sensor nodes should be deployed at a node density higher than the critical node density value which depends on the sensing range of the sensor nodes and can be obtained by the simulations.

In order to provide the full coverage with minimal energy consumption, the placement problem of sensor nodes into a monitored area and their organization are examined in [27]. Specifically, authors develop a heuristic that organizes the randomly deployed sensor nodes into mutually exclusive sets where the members of each of those sets completely cover the monitored area. Only one such set is active and consumes power at any moment. After a specified interval another set is activated, while the first one is deactivated. In one round all sets are used, and then the whole process repeats until the sensor nodes are out of power. The lifetime of the whole system directly corresponds to the number of allocated sets. Therefore the goal of the algorithm is to maximize the number of the sets. By using only a subset of sensor nodes at each moment, significant energy savings is achieved while fully preserving coverage.

**b. Directed Deployment**

Directed deployment strategy is mainly used in friendly, accessible and smaller sensor fields. There has been quite big range of work is done with this strategy.

In [18] authors present two-fault tolerant node placement algorithms for sensor deployment that provide sufficient area coverage. Sensor field is modeled

as 2D grid of points and their algorithms situate the sensor nodes to the minimum covered points of the grid. Their sensor deployment strategy reduces the number of sensor nodes significantly compared to random deployment while taking care of the failures of the sensor nodes in the field.

In [19][30] an optimization problem on sensor placement is formulated to provide sufficient grid coverage of the sensor field where two polynomial-time algorithms are presented to find out the optimum number of sensors and to place them such that the maximum coverage of the sensor field is achieved. The proposed scheme is for fixed sensor nodes and runs better for sensor fields that have obstacles. For the case that there are some obstacles that can hinder communication between nodes in the field, the knowledge of the terrain is required before deployment.

Sensor placement is formulated as an optimization problem and then solved with ILP in [20]. This approach deals with sensor fields that have various sensor types different in costs and ranges. It finds out the types and the locations of sensor nodes for maximum area coverage of the sensor field in terms of cost minimization. Necessity of using divide and conquer strategy due to complexity of the proposed algorithm makes this approach infeasible for large problem instances.

A coding theory framework for target location in distributed WSNs is presented in [23]. Authors provide coding-theoretic bounds on the number of sensors and present methods for determining their placement in the sensor field. This approach represents the sensor field as a grid of points and sensor nodes are selectively placed on a subset of these grid points for area coverage. They used the framework of identifying codes to determine sensor placement for unique target location, which refers to the problem of pin-pointing a target at a grid point at any instant in time.

In [31], self-organizing algorithms that exploit the redundancy in WSNs for area coverage are presented. To conserve the overall battery power of the network, algorithms select a small subset of sensor nodes that is sufficient to process a given query. The authors primarily focused on reducing the communication by using enough number of sensor nodes while the other nodes are put to sleep modes.

The Art Gallery Problem (AGP) [32][33] is intended to determine the minimum number of observers needed to cover the interior of an art gallery room such that every point is covered by at least one observer. This problem can be considered as the complete area coverage. Several kinds of AGP have been examined in the literature, embracing mobile guards, exterior visibility, and polygons with holes. An optimal solution of the problem in 2D was given and 3D case was shown to be NP hard and heuristics for solving 3D case were presented by using Delaunay triangulations in [33].

### c. Autonomous Deployment

Autonomous deployment is used for mobile sensor nodes especially when we do not have any knowledge of the interior of the area that we want to monitor. After deploying mobile sensor nodes to any initial locations they move themselves to the sensing locations in the sensor field.

To the best of our knowledge [34] and [35] is the first published works in the literature which use autonomous deployment in WSNs. Authors describe an incremental deployment algorithm for mobile WSNs in which sensor nodes are deployed one at a time into an unknown environment. Sensor nodes are limited to the retain line of sight with one another due to the need to localize sensor nodes while maximizing the network coverage. Each sensor node makes use of the information gathered by the previously deployed sensor nodes to determine its optimal location. This algorithm is both incremental and greedy. The algorithm is

greedy in the sense that it attempts to determine the location for each sensor node that will produce the maximum increase in the area coverage of the sensor network.

In [17] a distributed and scalable solution to the studies mentioned in the above paragraph is presented by using a potential-field-based approach. Potential-field techniques [36] are widely used for the tasks such as local navigation and obstacle avoidance in the literature. In this work, potential-field-based approach for deployment is described, in which sensor nodes are treated as virtual particles, subject to virtual and viscous forces. Virtual forces repel nodes from each other and from obstacles, and ensure that an initial, compact configuration of nodes will quickly spread out to maximize the area coverage of the WSN. Viscous force is used to ensure that the network will ultimately come to a state of static equilibrium.

In [21] a "virtual force algorithm" (VFA) is presented as a sensor deployment strategy to enhance the area coverage of the WSN. Initially sensor nodes deployed randomly over a field which is represented by a 2D grid. In the sensor field, each sensor node behaves as a "source of force" for all other sensor nodes. This force can be either positive (attractive) or negative (repulsive). If two sensors are placed too close to each other, the "closeness" being measured by a pre-determined threshold, they exert negative forces on each other. On the other hand, if a pair of sensors is too far apart from each other, they exert positive forces on each other. The key idea of this approach is that the coverage provided by a random deployment can be improved by using a force-directed algorithm. For a given number of sensor nodes within a cluster in cluster-based WSNs, proposed algorithm attempts to maximize the sensor field coverage using a combination of these attractive and repulsive forces. The cluster head is responsible for executing the VFA algorithm and managing the one-time movement of sensors to the desired locations. During the execution of the force-directed algorithm, sensor nodes do not physically move but a sequence of virtual motion paths is

determined for the randomly-placed sensor nodes. Once the effective sensor node positions are identified, a one-time movement is carried out to redeploy the sensor nodes at these positions.

In [25] a self-deployment algorithm for mobile WSNs is presented. The authors are interested in a deployment strategy that maximizes the area coverage of the network with the constraint that each of the sensor nodes has at least $K$ neighbors, where $K$ is a user-specified parameter. This user specified parameter is called as "node degree" and brings flexibility to the algorithm. The proposed algorithm is based on artificial potential fields which is distributed, scalable and does not require a priori map of the environment. The pair-wise interaction between nodes is governed by two different kinds of forces. One of them causes the nodes to repeal each other to improve their coverage and the other is an attractive force that prevents the nodes from losing connectivity. By using a combination of these two forces every sensor nodes tries to maximize its coverage while maintaining the required number of neighbors.

### C. WHAT IS THE RATIONALE FOR USING DISTRIBUTED ALGORITHMS IN WSNs?

WSNs are based on the collaborative effort of vast number of tiny sensor nodes embedded in the phenomenon to be observed [1]. The sheer number of these sensor nodes makes global broadcasting undesirable because of the low power consumption requirement of WSNs. In WSNs centralized solutions are unacceptable for the following reasons [3][5][37]:

- They provide a single point of failure.
- They are energy inefficient.
- They do not scale well for large networks.

In distributed algorithms sensor nodes only interact with the nodes within some neighborhood while they achieve a global objective. Since sensor nodes themselves are physically distributed, it is not unnatural to design WSNs using distributed algorithms. The attractive features of distributed localized algorithms can be summarized as follows [5][11][37]:

1. Each sensor node communicates only with its neighboring nodes. The communication overhead scales well with the network size. Control decisions are based on the interactions with these neighboring nodes without any global system knowledge.

2. These algorithms are robust to network partitions and individual node failures.

As a conclusion to above discussion, distributed algorithms should be preferred to centralized algorithms in WSNs. Therefore we design distributed algorithms to solve the 3D space coverage problem of underwater wireless sensor networks.

# III. THE DISTRIBUTED 3D SPACE COVERAGE SCHEMES FOR UNDERWATER WIRELESS SENSOR NETWORKS

## A. PRELIMINARIES

Before we delve into the details of our distributed scheme, we first introduce our coordinate system model, formal definition of our problem and the constraints.

### 1. Coordinate System Model

We use a 3D grid based coordinate system to denote the node locations. According to our coordinate system, a sensor space is a 3D space that covers all the sensor nodes. We accept a corner of the sensor space as the *origin* of our coordinate system. Since sensor nodes may move along with current or tide, the *origin* point should be selected according to the expected places of the nodes for the intended sensing period. After assigning the *origin* point, we partition the whole sensor space into the cubes $r$ in size. $r$ is called the *resolution distance* which is the edge length of a unit cube that sensor space is partitioned into. Nodes find out their locations based on the *origin* point and the *resolution distance* (*r*). Our coordinate system is shown in Figure 5.

The address of a node is the cube that it is in. The number of bits to denote such an address depends on parameters width $w$, length $l$, and depth $d$ of the sensor space. The number of bits to denote $x$, $y$, and $z$ coordinates of a node is as follows;

$$n_x = \lceil \log_2 x \rceil \quad \text{where } x = \frac{w}{r} \tag{6}$$

$$n_y = \lceil \log_2 y \rceil \quad \text{where } y = \frac{l}{r} \tag{7}$$

$$n_z = \lceil \log_2 z \rceil \quad \text{where } z = \frac{d}{r} \tag{8}$$

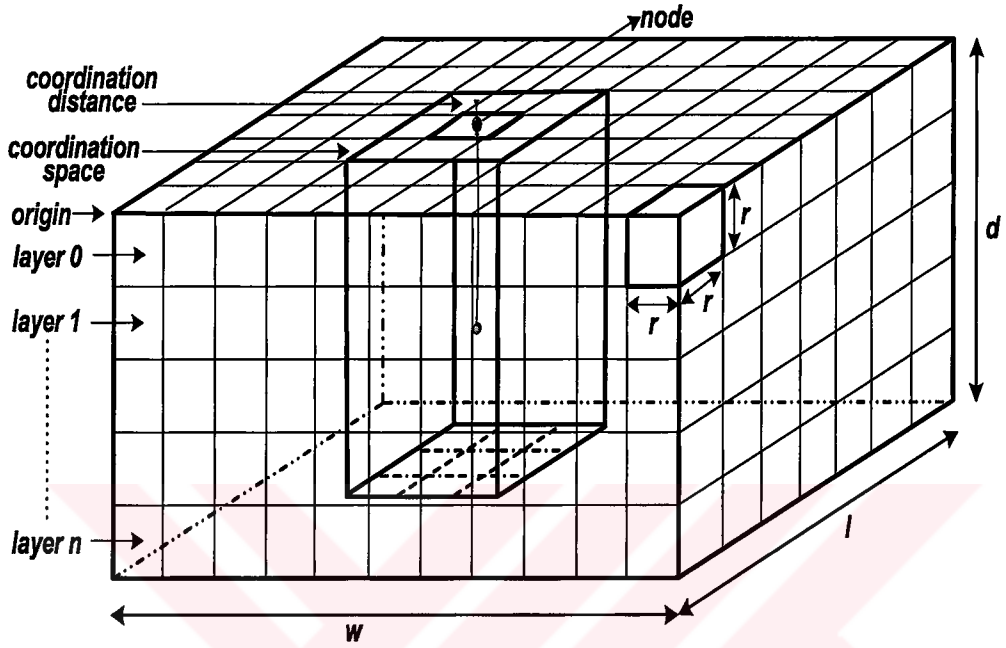



*Figure 5. The coordinate system for a sample sensor space with coordination distance ($\alpha$) = 1.*

For example, the length of the address field for a sensor space of 1000 m in width, 500 m in length and 100 m in depth with a *resolution distance* of 10 m, is as follows;

$$n_x = \lceil \log_2 100 \rceil = 7$$

$$n_y = \lceil \log_2 50 \rceil = 6$$

$$n_z = \lceil \log_2 10 \rceil = 4$$

Address of a node at geographic coordinates (36128, 41256, 50) when origin coordinates $x = 35500$ and $y = 41000$ is presented below.

$$C_x = \left\lfloor \frac{36128 - 35500}{10} \right\rfloor = 62,$$

$$C_y = \left\lfloor \frac{41256 - 41000}{10} \right\rfloor = 25 \text{ and}$$

$$C_z = \left\lfloor \frac{50}{10} \right\rfloor = 5$$

Thus we can address this node by using 17 bits as;

$$C_{node} = (0111110, 011001, 0101)$$

In addition to the *resolution distance* we use another parameter $\alpha$, the *coordination distance*, shown in Figure 5. *Coordination distance* indicates the distance in the number of neighboring cubes for a node to coordinate its depth. While a node arranges its depth, it exchanges information with the nodes in the *coordination distance.*

We also identify each cube along with the z axis from sea surface to maximum depth with a layer number. Hence all the cubes at the same depth construct a layer, i.e. all the cubes between depth *0* and depth *r* construct layer 0.

## 2. The Formal Definition of Our Problem

Our goals are to;

a. Maximize *coverage efficiency* ($\delta$)

$$\text{where} \quad \delta = \frac{n_c}{n_n} \tag{9}$$

$n_c$ is the number of the cubes covered by at least one sensor node and $n_n$ is the number of nodes in the sensor space. This aims to provide maximal coverage.

b. Maximize *average distance* ($\theta$)

$$\text{where} \quad \theta = \frac{\sum_{i=1}^{k} e_i}{k} \tag{10}$$

In Equation 10, $e_i$ is the distance between two nodes in node pair $i$. If there are n nodes in a sensor field we can have as many as $k$ node pairs that can be found by Equation 11.

$$k = \frac{n_n \cdot (n_n - 1)}{2} \tag{11}$$

This goal aims to reduce the possibility that all nodes are concentrated on certain depths. Thus we hinder the existence of corridors so that an intruder cannot find an uncovered depth.

### 3. Constraints

Our goals are subject to the following constraints;

*a.* We cannot move nodes in X and Y axes. Nodes are randomly deployed. We can arrange only the depths of nodes.

*b.* Nodes may move with currents or tide. They most probably move in the same direction with the same speed. Therefore their relative locations to each

other normally do not change. However if there are currents in different directions for different depths, then even the relative locations of nodes may change.

*c.* Our scheme needs to be distributed. We cannot rely on a central node that knows the up to date locations of each node and arranges the depth of each node centrally.

## B. THE DISTRIBUTED 3D SPACE COVERAGE SCHEMES

In this Section we introduce two different versions of our distributed scheme. For the scheme explained in section III.B.1, the sensor nodes must be location aware. The scheme explained in section III.B.2 is the version of our scheme which is adapted for the sensor networks made up of non-location aware sensor nodes.

### 1. The Distributed 3D Space Coverage Scheme For Sensor Networks Consist Of Location Aware Nodes

In this scheme, the nodes coordinate their depths with the other nodes in their *coordination space* which is the 3D space that include all the cubes whose $x$ and $y$ coordinates satisfy the following conditions:

$$x_i \le x_n + \alpha \le x_i + 2\alpha$$
$$y_i \le y_n + \alpha \le y_i + 2\alpha \qquad (12)$$

where $\alpha$ is *coordination distance*, $x_n$ and $y_n$ are the coordinates of a given cube and $x_i$ and $y_i$ are the coordinates of the node that the *coordination space* is calculated for. This is also depicted in Figure 5. A node exchange information with the nodes in its *coordination space* and find out an appropriate depth for itself by running our distributed algorithm and then tunes itself to that depth. The control packet formats and the flow diagram of the algorithm is shown in Figure 6 and 7 respectively.

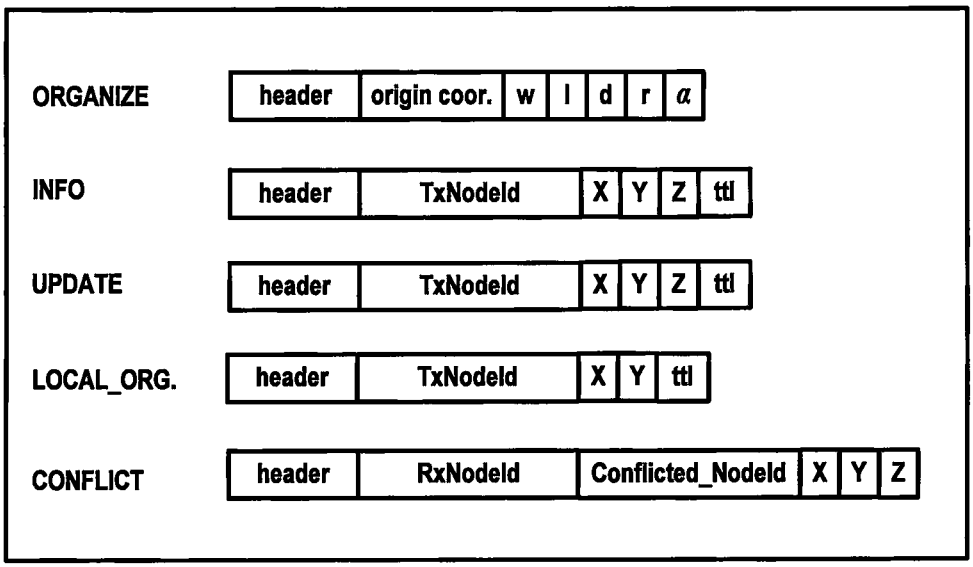



*Figure 6. Control packet formats.*

After the deployment, a global *organize* packet is broadcasted by the *cbuoy*, i.e. the sink. *Organize* packet contains the coordinates of the origin point, width (w), height (h) and depth (d) of the sensor space, *resolution distance* (*r*) and *coordination distance* ($\alpha$) as shown in Figure 6. After receiving *organize* packet nodes calculates their $x$ and $y$ coordinates according to the *origin point* and *resolution distance (r)* parameter as we stated before. Please note that from now on all the coordinates are given according to our coordinate system described in Section III.A.1. When a node receives an *organize* packet, it listens to the *info* packets of neighbor nodes in its *coordination space* for a randomly selected time period. *Info* packet contains the coordinates of the node that sends *info* packet, i.e., $x$, $y$ and $z$ coordinates. Therefore nodes learn the coordinates of neighbors as they receive *info* packets and store these data in their *neighbor table*. An instance of *neighbor table* is illustrated in Table 1. The records in this table are ordered according to $z$ value that is the depth of the related node.

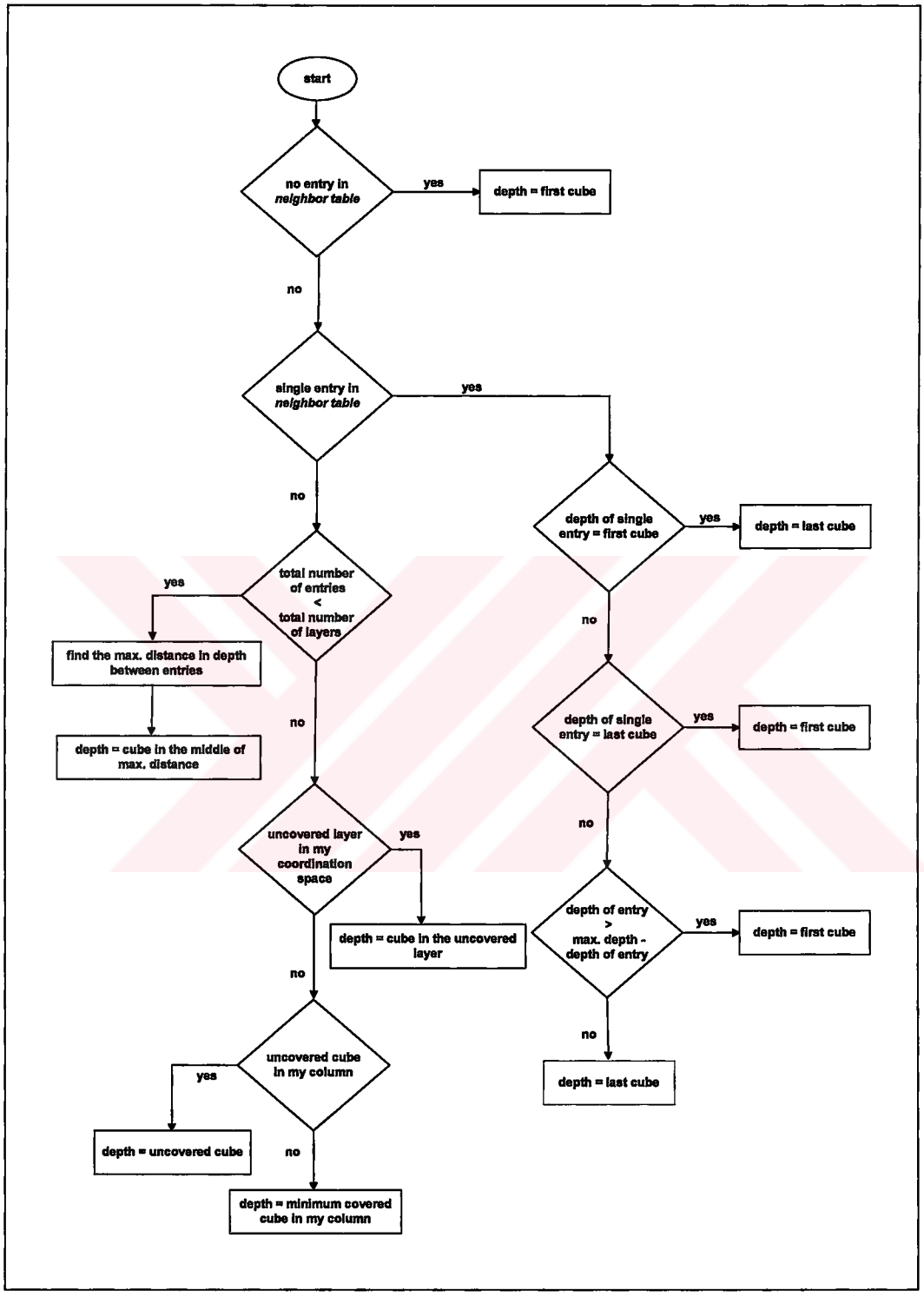



*Figure 7. The flow diagram of the distributed algorithm for sensor networks consists of location aware nodes.*

*Table 1. An instance of a neighbor table for location aware nodes.*

| NODE ID | X | Y | Z |
|---|---|---|---|
| 1245243 | 62 | 25 | 0 |
| 1245632 | 61 | 23 | 2 |
| 1243672 | 63 | 22 | 5 |
| 1246342 | 60 | 24 | 9 |

After waiting a random time period for *info* packets, the node finds out an appropriate depth for itself based on the data available in its *neighbor table.* Please note that the node may not hear from all of its neighbors in its *coordination space* by that time. Basically the node checks its *neighbor table* and runs the algorithm shown in Figure 7 to calculate its depth and tunes itself to the calculated cube (the middle point of the cube in depth axis) and then sends an *info* packet to inform its neighbors. When its uniformly selected time period expires it checks the data stored in its *neighbor table.* There can be three cases that a node encounters while calculating its depth;

*a. There is no entry in the neighbor table:* If there is no entry in the *neighbor table* node selects the first cube as its depth.

*b. There is only one entry in the neighbor table:* If there is only one entry in the *neighbor table* node checks the depth of the entry and tries to maximize the *average distance* ($\theta$). If the depth of the entry is the first cube it selects the last cube as its depth. If the depth of the entry is the last cube it selects the first cube as its depth. But the depth of the entry can be different from the first cube or the last cube. Under this circumstance, if the depth value of the single entry is bigger than the difference between the maximum depth and the depth of

the entry, node selects the first cube as its depth otherwise node selects the last cube.

*c. There are more than one entry in the neighbor table:* If there is more than one entry in the *neighbor table* node examines this situation in two cases:

*1. Total number of entries in the neighbor table is smaller than the total number of layers:* As we stated before, each cube along with the z axis from sea surface to maximum depth is identified with a layer number and all the cubes at the same depth construct a layer. If the total number of entries in the neighbor table is smaller than the total number of layers node finds the maximum distance in depth between the entries and then arranges its depth to the cube in the middle of that distance.

*2. Total number of entries in the neighbor table is not smaller than the total number of layers:* Although the total number of entries in the *neighbor table* is equal or greater than the total number of layers there can be an uncovered layer in the *coordination space* of the node. Therefore node checks if there is an uncovered layer in its *coordination space*. If there is an uncovered layer nodes selects the cube in this uncovered layer as its depth. If there is not an uncovered layer, in other words there is at least one sensor node in every layer, node checks its column (all the cubes along with the z axis from the cube where it is in). If there is an uncovered cube in its column node selects this uncovered cube as its depth. If all the cubes in its column are covered, node selects the cube which is covered by minimum number of nodes in order to maintain a homogeneous node distribution to all the layers.

For example, in a sensor field that has 100 m depth the appropriate cube (depth) for the node that has the *neighbor table* shown in Table 1 will be "7" when *resolution distance r* is 10 meters. According to these parameters the sensor space has 10 layers. When we look at the *neighbor table* it can easily be seen that

the total number of entries is smaller than the total number of layers. For such a case the node finds the maximum distance in depth between the entries and then tunes its depth to the cube in the middle of that distance. The distance between the nodes in the $5^{th}$ and $9^{th}$ cubes is the maximum comparing to the other distances between the nodes, therefore the node locates itself in the middle of that distance which is the $7^{th}$ cube.

Nodes may change their locations due to some reasons such as wind, current etc. When a node leaves its cube it sends an *update* packet that contains new location data. Thus neighbor nodes of the migrating node learn that the node left its cube. When a node receives an *update* packet it replies with an *info* packet if the sender node of the *update* packet is in its *coordination space.* Thus the migrating node learns the coordinates of new neighbors and then arranges its depth according to new *neighbor table* information.

In the reception of an *update* packet, the neighbor nodes in the previous *coordination space* of the migrating node decide if there is a need for local organization. This decision is taken according to *local coverage density* ($\delta$) value for their *coordination space* described in Section III.A.2. If $\delta$ value for a node in the previous *coordination space* of the migrating node smaller than the pre-defined threshold value, it sends a *local_org* packet and a local organization process which is similar to global organization starts. Neighbor nodes of the sender node of *local_org* packet will participate the local organization. These participating nodes listen to the *info* packets of their neighbor nodes in their *coordination space* for a randomly selected time period as they do in global organization. They store the information included in the *info* packets in their *neighbor tables*. When their randomly selected time period expires they run the distributed algorithm with the new *neighbor table* information to calculate their new depth. After they calculate their depths they send an *info* packet to inform their neighbors.

The *origin* coordinates should be selected such that the sensor space is covered for the required time period as explained in Section III.A.2. Nodes may move out of the sensor field in time which means the nodes cannot be addressed by using the current parameters such as origin coordinates, width (*w*), length (*l*), and depth (*d*). In this case, a new *organize* packet that contains new values for these parameters is broadcasted by the *cbuoy* and a new global organization restarts.

As we consider the hidden terminal problem two nodes in the same *coordination space* may use the same depth. When a node notices such a case from its *neighbor table* and there is an uncovered cube in its *coordination space,* it sends a *conflict* packet that contains conflicted node data to one of these nodes to adjust its depth. The receiver node stores the data of *conflict* packet in its *neighbor table* if the conflicted node is in its *coordination space* and runs the algorithm with its new *neighbor table* information. After calculating its depth it rearranges its depth to the calculated depth and sends an *info* packet to inform its neighbors.

The prosperity of our algorithm depends on the control packets that are sent by each sensor node to be received by every neighbor node in its *coordination space.* To achieve this we append *time to live* (*ttl*) field to *info*, *update* and *local_org* packets. When a node takes one of these control packets from any node in its *coordination space*, it decreases the *ttl* field value of the packet and forwards it if *ttl* value is greater than 0. Since energy consumption is critical in wireless sensor networks, the number of the transmitted packets is very important and *ttl* value must be selected carefully. Therefore we develop a method for evaluating *ttl* field values of control packets based on the distance between the nodes in the same *coordination space* and the transmission range of the sensor nodes.

For the worst case scenario, Equation 13 gives the maximum distance $d_{max}$ between the sender node of *info* and *local_org* packets and the receiver nodes in the same *coordination space.* In Figure 8, bird's eye view of sensor field (sea surface) is presented to illustrate the worst case scenario distances between *node a* and its neighbor nodes for various *coordination distances* ($\alpha$).

$$d_{\max} = \sqrt{2}\, r(\alpha + 1) \qquad (13)$$

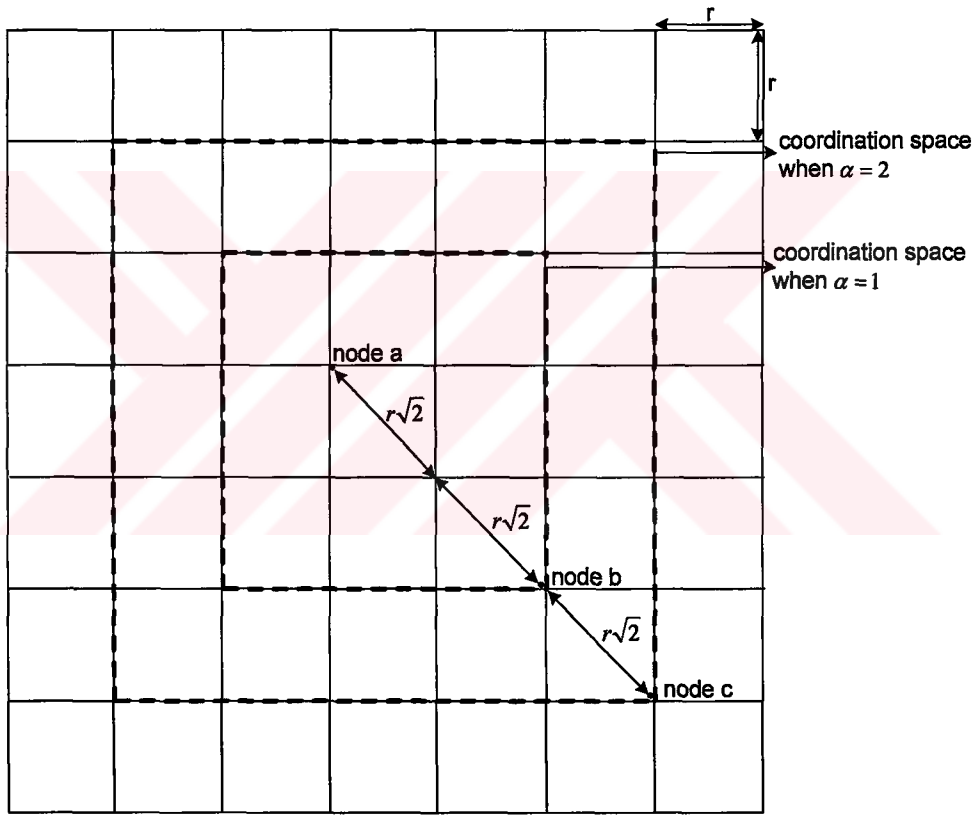



*Figure 8. The worst case scenario distances between node a and its neighbor nodes for various coordination distances ($\alpha$).*

As we stated before when a node change its location due to wind, current etc. it transmits an *update* packet to inform its neighbor nodes. When a node needs to send an *update* packet, both the nodes in the previous *coordination space* and

the nodes in the new *coordination space* must receive this packet. In Figure 9 it is illustrated that *node a* leaves its cube and the grey arrowed line shows the new location of *node a.* As it can easily be seen from Figure 9, the maximum distance $d_{max}$ that can be between the sender node of the *update* packet and the intended receiver nodes will be;

$$d_{\max} = \sqrt{2}\, r(\alpha + 2) \tag{14}$$

Therefore *ttl* field value for control packets is evaluated by the following Equation where $t_x$ is the transmission range of a sensor node.

$$ttl = \left\lceil \frac{d_{\max}}{t_x} \right\rceil \tag{15}$$

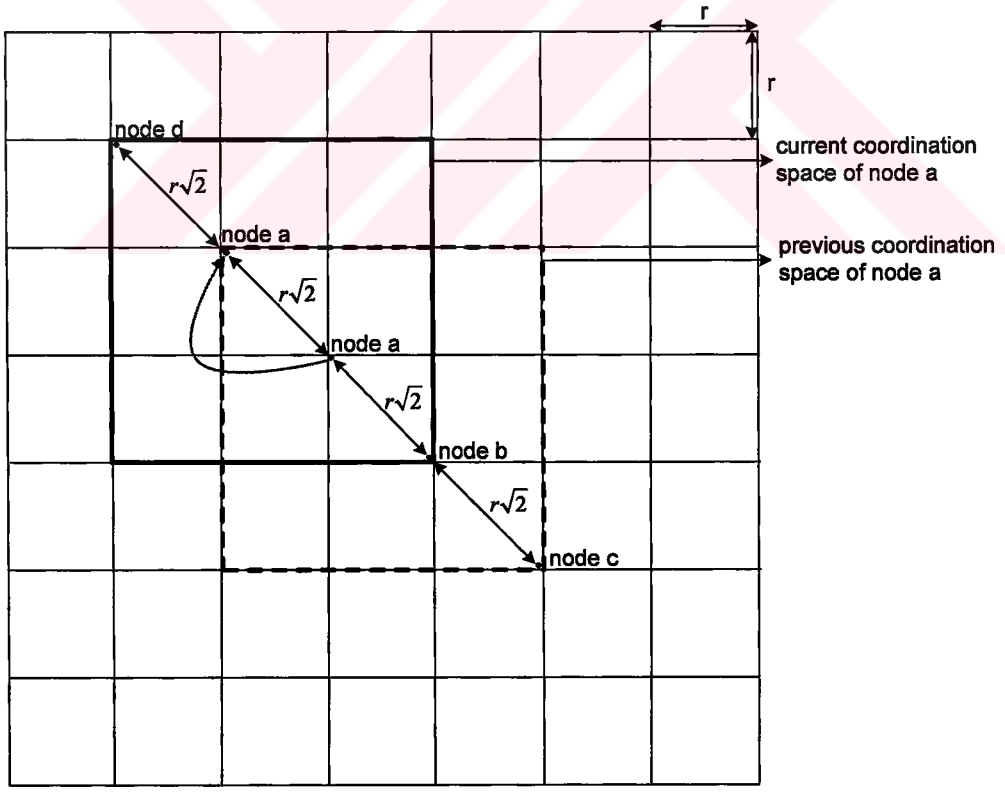



*Figure 9. The worst case scenario distances between node a and its current neighbor nodes and previous neighbor nodes when coordination distance = 1.*

The parameters, *resolution distance (r)* and *coordination distance ($\alpha$)* are the key factors of our algorithm. Bigger $\alpha$ value brings more neighbor nodes to exchange information and causes more control message overhead. On the other hand bigger $r$ value means lower resolution. We examine the influence of these parameters on the performance of our algorithm in detail during our experiments.

Our distributed algorithm given in Figure 7 is based on the control packets shown in Figure 6. Thus a dynamic structure is maintained and sensor network is robust to individual sensor failures and any significant topology changes.

Our distributed algorithm necessitates location awareness of nodes. Practical techniques to find out node locations presented in [38][39][40][41][42]. Therefore assuming location awareness of surface buoys is acceptable for our sensor network. Since we can lower the sensor nodes to desired depths, exact 3D locations of the sensor nodes can easily be found by appending this depth ($z$) value to the $x$ and $y$ locations of the surface buoys obtained by any of these techniques.

### 2. The Distributed 3D Space Coverage Scheme for Sensor Networks Consist of Non-Location Aware Nodes

This is the adapted version of our scheme for the sensor networks made up of non-location aware sensor nodes. Different from the location aware case, when nodes are not location aware they cannot find out their coordinates. That's why they cannot determine their neighbors according to their locations. However they can easily find out their neighbors based on hop distances. Therefore in this version of our scheme we use *coordination distance* ($\alpha$) parameter to denote hop distance. According to this adapted version of our scheme a node exchange information with $\alpha$ hop neighboring nodes and find out an appropriate depth for itself by running our distributed algorithm and then tunes itself to that depth. A

sample sensor network consists of non-location aware nodes is illustrated in Figure 10.

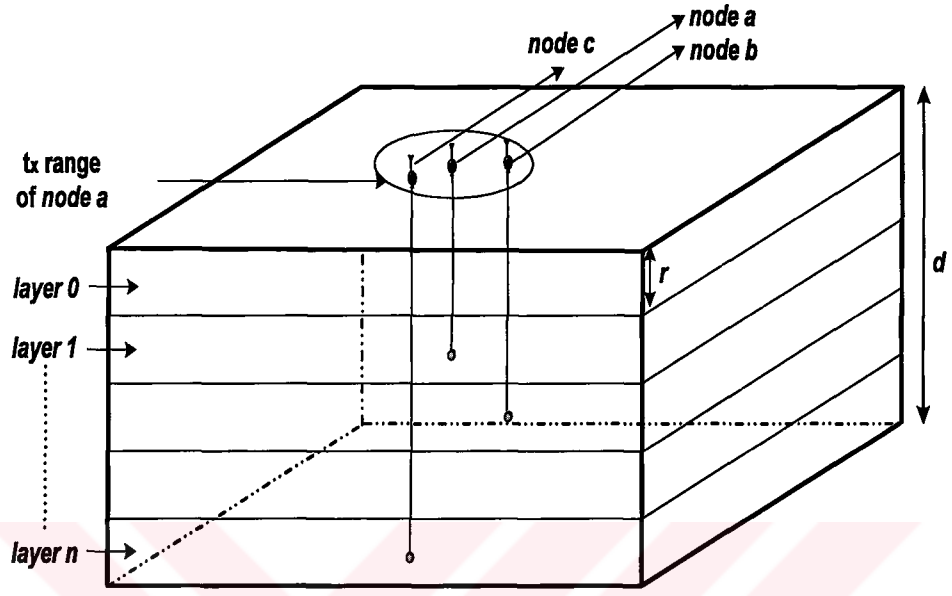



*Figure 10. A sample sensor space for a sensor network consist of non-location aware nodes*

Non-location aware version of our distributed scheme is also based on some control packets as the location aware version of our scheme is. The control packet formats are shown in Figure 11. According to this scheme, nodes initialize *ttl* value in their *info* packets with the *coordination distance* ($\alpha$). Every node receives these packets decrements the *ttl* field and relay them if *ttl* is greater than 0. For example when we the *coordination distance* ($\alpha$) is 1, *node a* exchange information with the *nodes b* and *c* as illustrated in Figure 10.

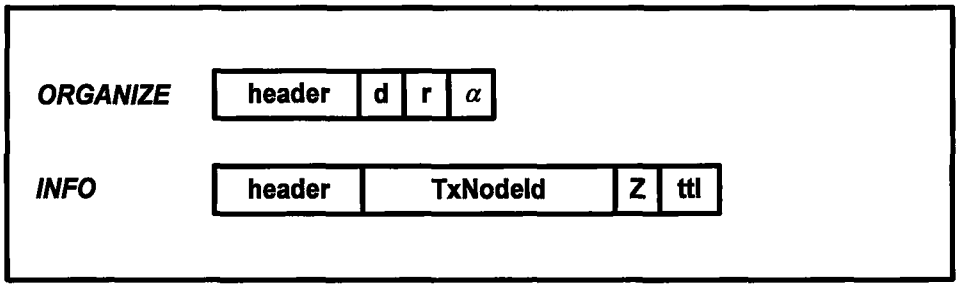



*Figure 11. Control packet formats.*

After the deployment, an *organize* packet is broadcasted by the *cbuoy*, i.e. the sink. *Organize* packet contains the depth of sensor space (d), *resolution distance* (*r*) and *coordination distance* ($\alpha$) as shown in Figure 11. When a node receives an *organize* packet, it listens to the *info* packets of its neighboring nodes for a randomly selected time period. *Info* packet contains the depth of the node that sends *info* packet, i.e $z$ coordinates. Please note that $z$ coordinate (depth) is calculated based on *resolution distance* parameter as explained in Section III.A.1. Thus nodes calculate their depth in layers as illustrated in Figure 10. Nodes learn the depths of their neighboring nodes as they receive *info* packets and store these data in their *neighbor table*. An instance of *neighbor table* is shown in Table 2. The records in this table are ordered according to $z$ value that is the depth of the related node.

*Table 2. An instance of a neighbor table for non-location aware nodes.*

| NODE ID | Z |
|---|---|
| 1245243 | 0 |
| 1245632 | 2 |
| 1243672 | 5 |
| 1246342 | 9 |

Similar to the location aware version of our scheme, after waiting a random time period for *info* packets, each node finds out an appropriate depth for itself based on the available data in its *neighbor table*. Please note that the node may not hear from all of $\alpha$ hop neighboring nodes by that time. Basically the node checks the available data in its *neighbor table* and runs our distributed algorithm and calculates its depth. The flow diagram of our distributed algorithm is shown in Figure 12. When its uniformly selected time period expires node checks the data stored in its *neighbor table*. There can be three cases that a node encounters while calculating its depth:

*a. There is no entry in the neighbor table:* If there is no entry in the *neighbor table* node selects the first layer as its depth.

*b. There is only one entry in the neighbor table:* If there is only one entry in the *neighbor table* node checks the depth of the entry and tries to maximize the *average distance.* If the depth of the entry is the first layer it selects the last layer as its depth. If the depth of the entry is the last layer it selects the first layer as its depth. But the depth of the entry can be different from the first layer or the last layer. Under this circumstance, if the depth value of the single entry is bigger than the difference between the last layer and the depth of the entry, node selects the first layer as its depth otherwise node selects the last layer.

*c. There are more than one entry in the neighbor table:* If there is more than one entry in the *neighbor table* node examines this situation in two cases:

*1. Total number of entries in the neighbor table is smaller than the total number of layers:* If the total number of entries in the *neighbor table* is smaller than the total number of layers node finds the maximum distance in depth between the entries and then arranges its depth to the layer in the middle of that distance.

*2. Total number of entries in the neighbor table is not smaller than the total number layers:* Although the total number of entries in the *neighbor table* is equal or greater than the total number of layers there can be layers which are not covered by its $\alpha$ hop neighboring nodes. Therefore node checks if there is an uncovered layer. If there is an uncovered layer nodes selects this uncovered layer as its depth. If there is not an uncovered layer, in other words there is at least one sensor node in every layer, node selects a layer which is covered by minimum number of nodes in order to maintain a homogeneous node distribution to all the layers.

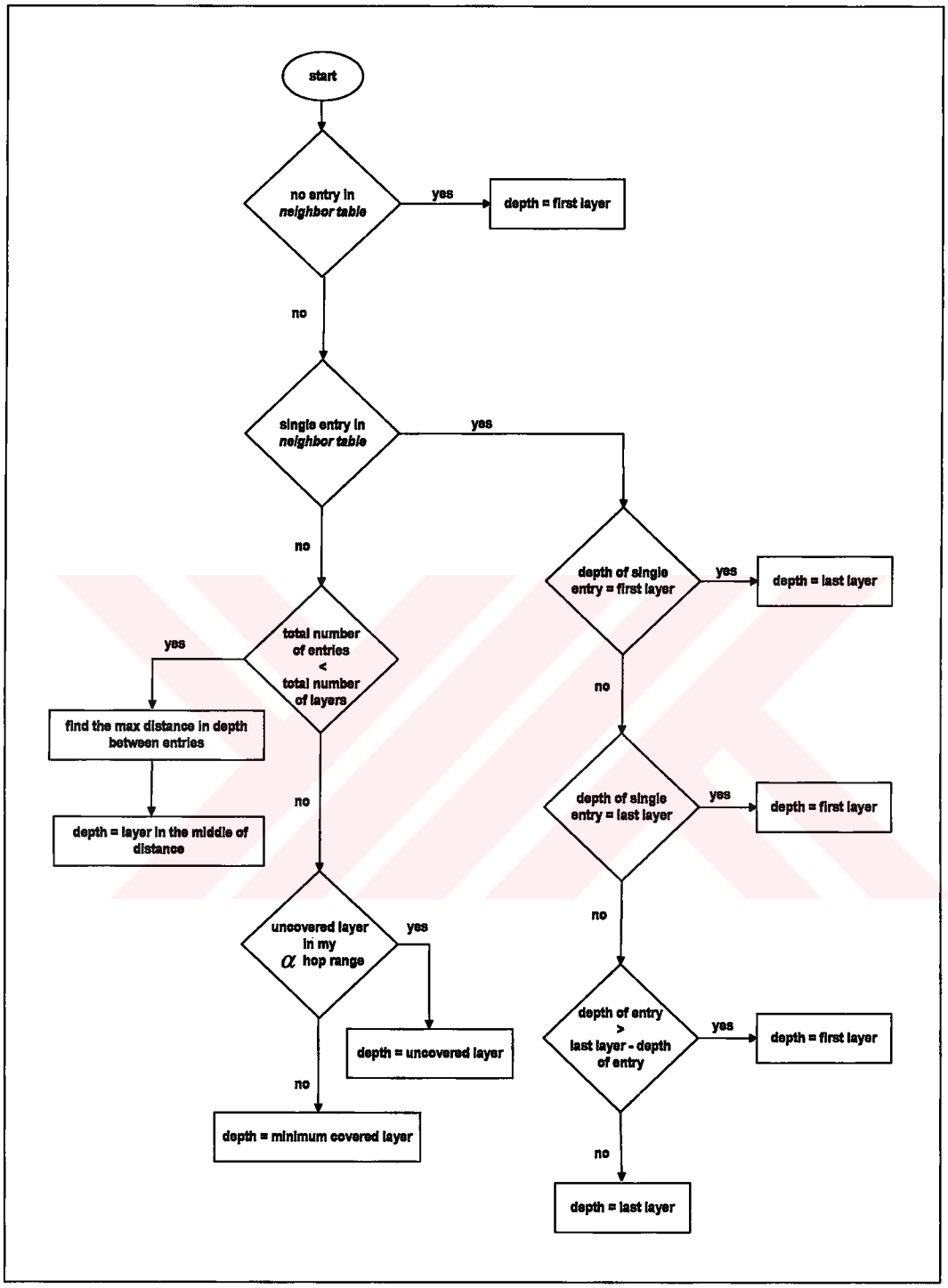



*Figure 12. The flow diagram of the distributed algorithm for sensor networks consists of non-location aware nodes.*

After calculating depth according to algorithm given in Figure 12 node tunes itself to the calculated layer (the middle point of the layer in depth axis) and then sends an *info* packet to inform its neighbors.

Similar to the scheme developed for the sensor networks consist of location aware nodes, the parameters, *resolution distance (r)* and *coordination distance ($\alpha$)* are also the key factors for this version of our algorithm. For the bigger values of $\alpha$, nodes exchange information with more neighboring nodes while they are calculating their depths. This causes more control message overhead. On the other hand bigger *r* value means lower resolution. In addition to the parameters, *resolution distance (r)* and *coordination distance ($\alpha$)* transmission range of nodes is another important factor for this version of our algorithm. In this algorithm, the number of neighboring nodes for a node mainly depends on the transmission range of it. When the transmission range of nodes increases the number of their neighboring nodes also increases. When their transmission range decreases they exchange information with less number of neighboring nodes while they are calculating their depths. We examine the impact of all these parameters on the performance of our algorithm in detail during our experiments.

### 3. Time Complexity Analysis of Our Distributed Algorithms

We use a mathematical notation called order of magnitude, or Big-O notation [43] for the time complexity analysis of our distributed algorithms. As we stated before, the records in the *neighbor table* of a node must be ordered according the *z* (depth) values. Therefore an efficient sorting algorithm such as *mergesort* that has the $O(N\log_2 N)$ time complexity can be used for sorting these records [43]. Except sorting, both of these distributed algorithms have single for loops and simple instructions such as addition, multiplication, division, comparison, and assignment. Hence our distributed algorithms have linear time complexity which can be denoted as O(N). Since the order of magnitude of an

algorithm is identified with the term that increases fastest relative to the size of the problem, the worst-case time complexity of our distributed schemes is $O(N\log_2 N)$.

# IV. PERFORMANCE EVALUATION

## A. SIMULATION SETUP

Before we begin to analyze our experiments we first explain the simulation setup we are using. We evaluated the performance of our distributed schemes by using ARENA program. In our simulations varying number of sensor nodes deployed randomly over square a field of 100×100 units in size and the maximum depth needed to cover under that field is 60 units. We accept that there are no currents in the sensor space. Therefore the locations of the nodes do not change after deployment. Sensor nodes randomly choose their waiting time between 1-60 seconds according to uniform distribution. The *resolution distance (r)* parameter is accepted 10 units, except when specifically stated otherwise. We accept the worst case scenario for the distance between the sensor nodes and use our approach for evaluating the *ttl* values of control packets. We assume that each sensor node has a radio module that has the transmission range of 10 units. After running our distributed algorithms sensor nodes select their depths between 0 and 60 units.

As our fundamental purpose is to maintain maximum coverage, *coverage efficiency* ($\delta$) and the *average distance* ($\theta$) between the nodes are the main criterion in our experiments. We compare our algorithm with a random depth selection model where depths of each node are selected randomly according to uniform distribution. We use the same seed for both our distributed schemes and random depth selection algorithm while we are randomly deploying the nodes to the sensor field.

We firstly present the experimental results for the distributed 3D space coverage scheme for sensor networks consist of location aware nodes in Section V.B. The performance evaluations of the distributed 3D space coverage scheme

for sensor networks consist of non-location aware nodes is presented in Section V.C.

## B. PERFORMANCE EVALUATION OF THE DISTRIBUTED 3D SPACE COVERAGE SCHEME FOR SENSOR NETWORKS CONSIST OF LOCATION AWARE NODES

### *1. Experiment-1: Sensitivity of Coordination Distance ($\alpha$) Over Coverage Efficiency ($\delta$)*

When we partition our sensor space (100×100×60) into the unit cubes which has the edge length of *resolution distance* (10), we obtain 600 unit cubes. Therefore we need at least 600 nodes to maintain full coverage of the sensor space when nodes are deployed to the sensor field (sea surface) such that there will be the total number of layers (6) of nodes in every column of the sensor space. In Figure 13 we compare the number of covered cubes with our scheme and random depth selection model for varying number of nodes deployed in the sensor space. For every number of sensor nodes our algorithm outperforms the random depth selection model.

As you remember that *coverage efficiency* ($\delta$) is the ratio of the number of covered cubes to the number of nodes in the sensor space. In Figure 14 the *coverage efficiency* of our algorithm and the random depth selection model is shown for varying number of nodes deployed in the sensor field. In general, our algorithm tries to cover all the uncovered cubes while maximizing the distance between the nodes. However, as a result of random deployment of nodes to the sensor field (sea surface), the number of nodes in the same column may exceed the total number of layers. For that reason a node may not find an uncovered cube to cover since all the cubes are covered by the neighbor nodes in its column. Therefore it arranges its depth to the cube which is covered by minimum number of nodes in its column. Unless the number of the sensor nodes in the same column

overruns the total number of layers, there will be no reduction in the *coverage efficiency* of our algorithm.

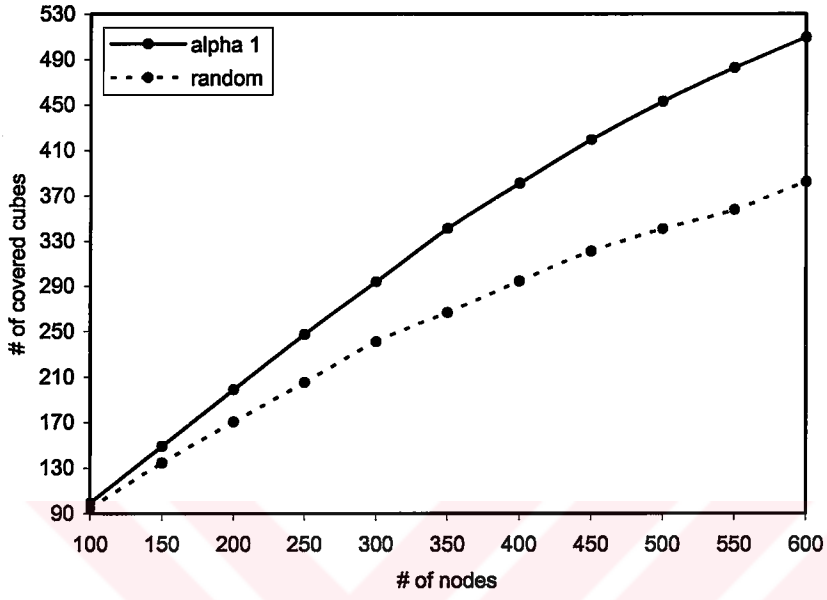



*Figure 13. Number of covered cubes with our scheme and random depth selection model.*

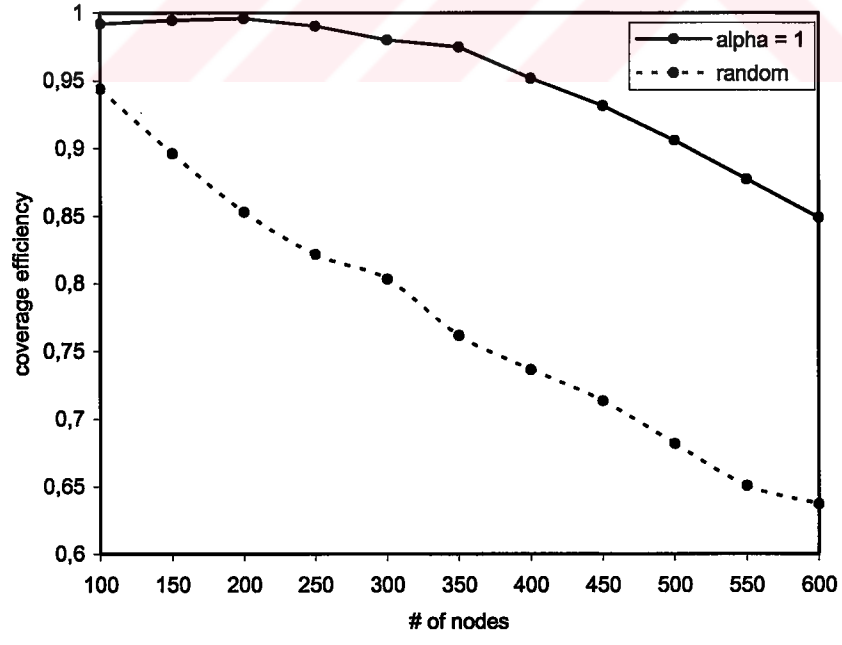



*Figure 14. Coverage efficiency ($\delta$).*

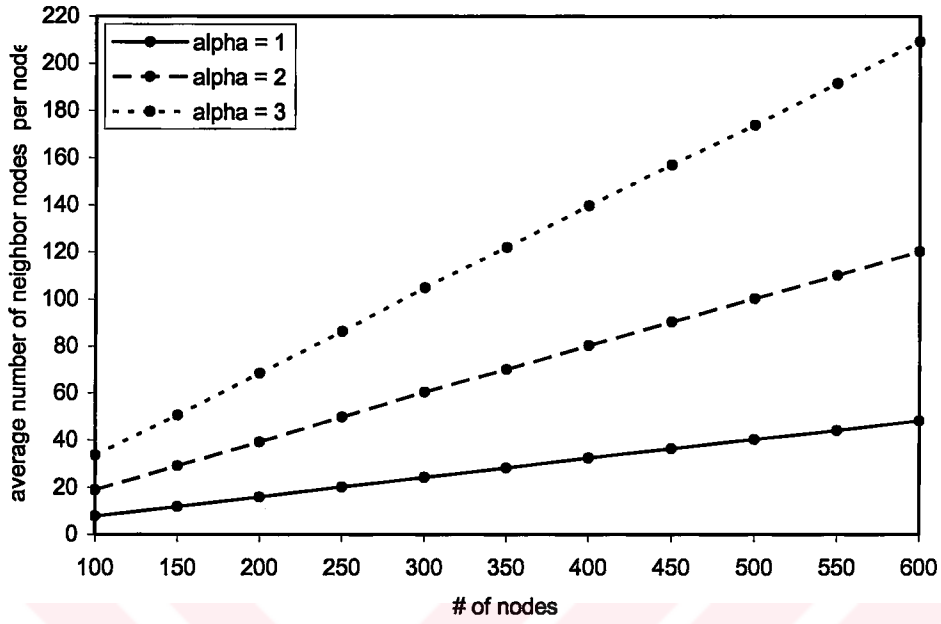



*Figure 15. Average number of neighbor nodes per node for varying coordination distances ($\alpha$).*

Average number of neighbor nodes per a node for varying *coordination distances* ($\alpha$) is shown in Figure 15. When the *coordination distance* increases, the *coordination space* of a node gets bigger. Therefore nodes exchange information with more neighbor nodes while they are calculating their depths. We compare the *coverage efficiency* of our algorithm for varying *coordination distances* in Figure 16. Since nodes negotiate with more neighbor nodes for the bigger *coordination distance* values, an increase in the *coordination distance* results in an improvement in the *coverage efficiency* as it can be seen in Figure 16. Since all the uncovered cubes in the *coordination distance* of a node will be covered after a certain number of nodes are deployed, our algorithm produces the same results for all the *coordination distance* values as expected.

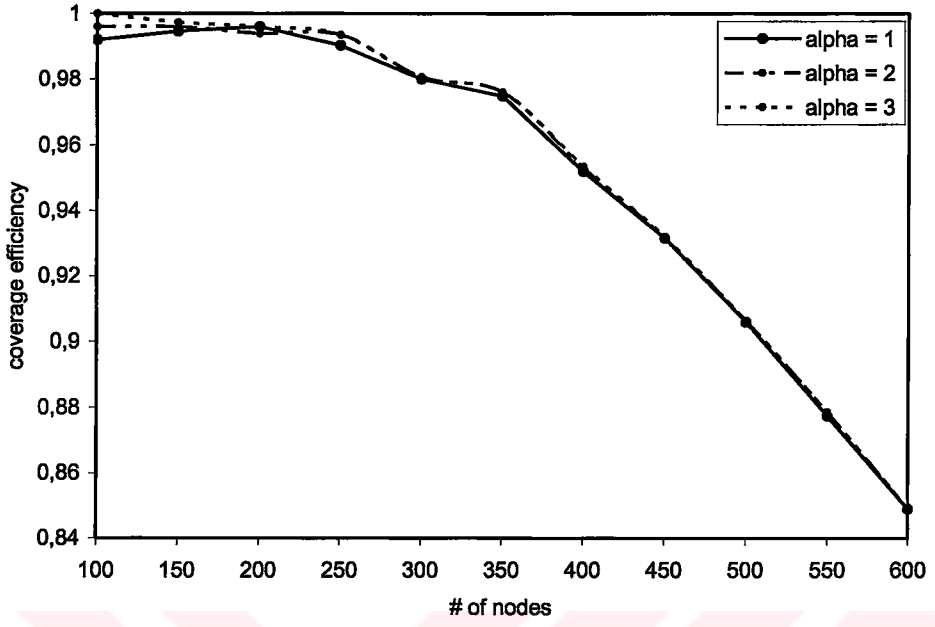



*Figure 16. Coverage efficiency ($\delta$) for varying coordination distances ($\alpha$).*

### *2. Experiment-2: Sensitivity of Coordination Distance ($\alpha$) Over Average Distance ($\theta$) Between The Nodes*

The locations of nodes on the $x$ and $y$ axis (sea surface) have a significant importance on *average distance* between the nodes. More separate deployment of nodes on the $x$ and $y$ axis will make a positive effect on the increase of *average distance.* On the other hand nearby deployment of nodes will make a negative effect on the increase of *average distance.* In order to increment the *average distance* nodes should be deployed as much separate as possible. But in our simulations we deployed nodes randomly according to the uniform distribution and the performance of our scheme is evaluated based on random deployment.

In Figure 17 and 18 *average distances* for our algorithm with varying *coordination distances ($\alpha$)* and random depth selection model are compared for various numbers of sensor nodes. Although the random depth selection model provides an even distribution of nodes in $z$ (depth) axis, as shown in Figure 17 our

algorithm outperforms it. When the *coordination distance* increases the number of neighbors of a node also increases as shown in Figure 15. A node try to select an uncovered layer in its *coordination space* when the total number of neighbors of a node exceeds the total number of layers, and ignore maximizing *average distance.* Therefore for the bigger values of *coordination distance* there will be a decrease in *average distance* until all the layers are covered. After all the layers are covered, nodes try to cover an uncovered cube in their column. Therefore after a certain number of nodes deployed, *average distance* is not sensitive to the changes in *coordination distance* and our algorithm produces the same results for all the *coordination distance* values as expected. The anomalies in the results that can be seen between 100 and 350 nodes deployed are due to the randomness in the deployment of nodes to the sensor field.

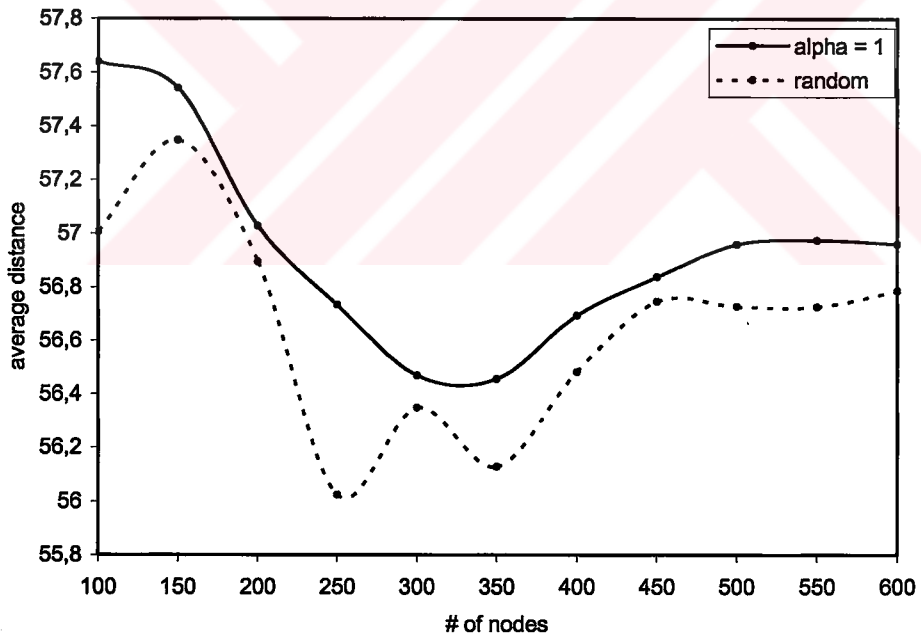



*Figure 17. Average distance (θ).*

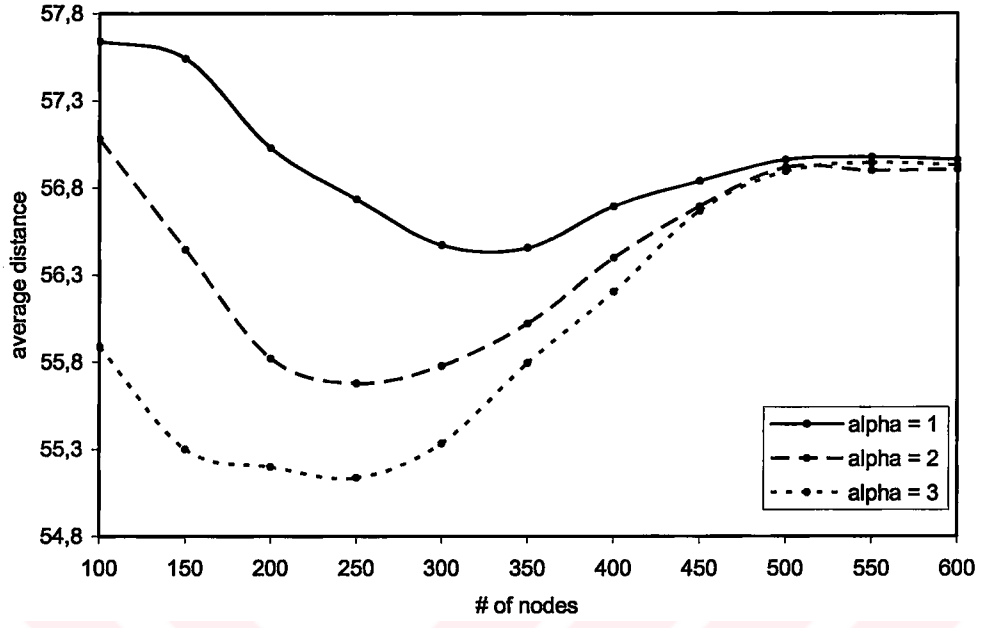



*Figure 18. Average distance ($\theta$) for varying coordination distances ($\alpha$).*

### *3. Experiment-3: Sensitivity of Coordination Distance ($\alpha$) Over The Average Number of Transmitted Control Packets Per Node*

As we stated in Section III.B.1, the prosperity of our algorithm depends on the control packets that are sent by each sensor node to be received by every neighbor nodes in its *coordination space*. In Figure 19 we compare the number of transmitted control packets per node for varying *coordination distances ($\alpha$)*. We use the worst case distance between nodes while we are evaluating *ttl* values of control packets. For the *coordination distances* 1, 2 and 3, the *ttl* values are calculated as 3, 5 and 6 respectively. Please note that a nodes relays the control packets if the sender node of the received packet is in its *coordination space*, otherwise node does not relay a packet even if the *ttl* field value of the packet is bigger than 0. Increase in *coordination distance* results more neighbor nodes to exchange information. In addition to that, when the total number of sensor nodes increases the number of neighbor nodes of a sensor node also increases and this results in more control packets to be forwarded. As we can see in Figure 19 the

number of control packets sent by a sensor node increases for the higher *coordination distance.*

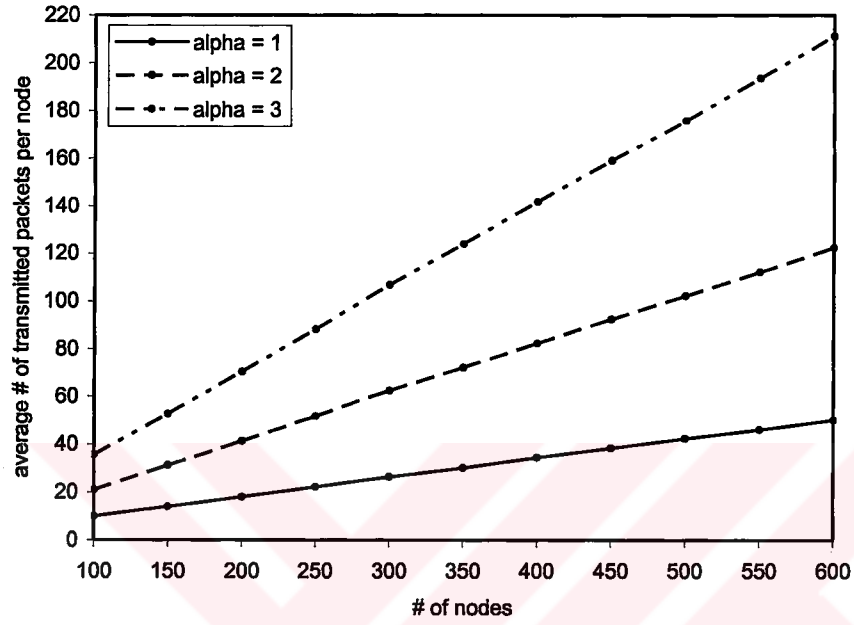



*Figure 19. Average number of transmitted packets per node for varying coordination distances ($\alpha$).*

### *4. Experiment-4: Sensitivity of Coordination Distance ($\alpha$) Over The Node Distribution to The Layers*

As we stated before the maximum depth of our sensor space is 60 units and we partition this space into the unit cubes which has the edge length of 10 units (*resolution distance*). Therefore we can assume that our sensor space consists of 6 layers. In Figure 20 sensor node distributions to these layers are shown for varying *coordination distances* of our algorithm and random depth selection model when 600 nodes are deployed to our sensor field. As shown in Figure 20, our algorithm maintains a homogenous distribution to all the layers for all *coordination distance* values. Thus we preclude the existence of corridors

(uncovered depth zones) and reduce the possibility that all nodes concentrated on certain depths.

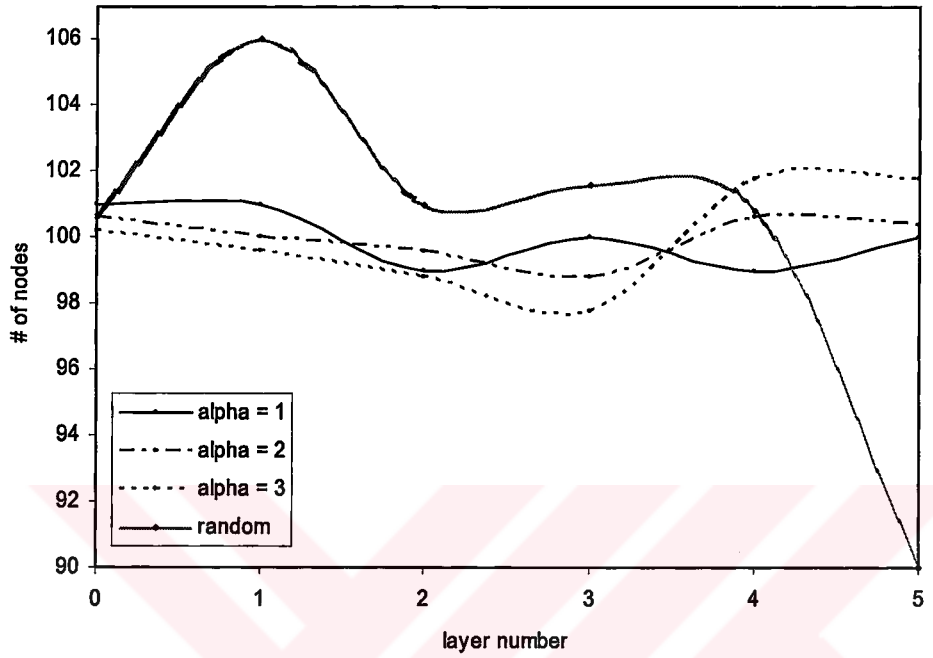



*Figure 20. The number of sensor nodes at each layer.*

### *5. Experiment-5: Sensitivity of Resolution Distance (r) Over Coverage Efficiency ($\delta$) and Average Distance ($\theta$) Between The Nodes*

As the *resolution distance* value increases, the volume of the unit cube increases. When we partition our sensor space with the unit cube which has the edge length of 20 (*resolution distance*) units we obtain 75 unit cubes and there will be only 3 layers. Therefore we use *node density* to compare the *coverage efficiency* of our algorithm with various *resolution distances*. Equation 16 gives the *node density* ($n_d$) of the network where $n_n$ is the number of nodes and $n_c$ is the number of cubes in the sensor space.

$$n_d = \frac{n_n}{n_c} \tag{16}$$

In Figure 21 coverage *efficiency* ($\delta$) of our algorithm for various *node densities* ($n_d$) are shown when *resolution distances (r)* parameter is accepted as 10 and 20 units respectively. When bigger *resolution distance (r)* values are used, the size of the unit cube that the sensor space partitioned into also gets bigger. As a result of random deployment of nodes to the sensor field (sea surface), for the bigger *resolution distances* when the *node density* increases the number of nodes in the same column exceeds the total number of layers. For that reason a node may not find an uncovered cube to cover since all the cubes are covered by the neighbor nodes in its column. Therefore it arranges its depth to the cube which is covered by minimum number of nodes in its column. Unless the number of the sensor nodes in the same column overruns the total number of layers, there will be no reduction in the *coverage efficiency* of our algorithm.

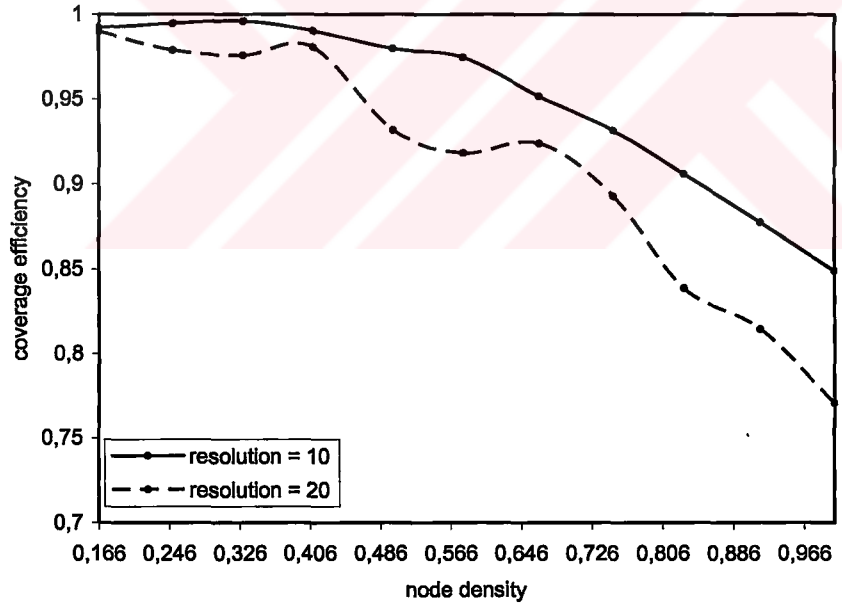



*Figure 21. Coverage efficiency ($\delta$) for varying resolution distances (r).*

In Figure 22 coverage *efficiency* ($\delta$) of our algorithm for various *resolution distances (r)* are compared when various number of nodes deployed to the sensor space. As we explain before, for the bigger *resolution distances* the dimensions of the unit cube and the layers increases. Therefore, for the higher

*resolution distance* values, when the deployed sensor nodes increases the number of sensor nodes per a cube rapidly increases and this results in an obvious reduction in the *coverage efficiency*. In Figure 22 the dotted curve fits the *coverage efficiency* curve of our algorithm when *resolution distance* is 20 units and can be denoted as follows:

$$y = 0,737e^{-0,003x} \tag{17}$$

where $y$ is *coverage efficiency and* $x$ is number of nodes in the sensor space.

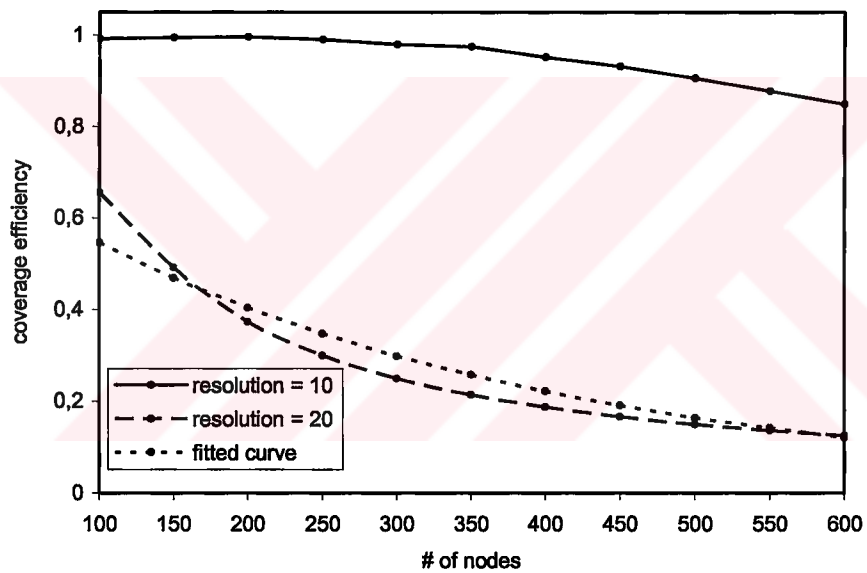



*Figure 22. Coverage efficiency ($\delta$) for varying resolution distances (r).*

In Figure 23 we compare *average distance* ($\theta$) of our algorithm for various *resolution distances (r)* when *coordination distance* ($\alpha$) parameter accepted as 1. As we describe above, when *resolution distance* parameter is accepted as 20 units the volume of the unit cube increase and the total number of layers of the sensor space decreases form 6 to 3. When the number of deployed sensor nodes increases the number of sensor nodes per a cube also per a layer rapidly increases. Therefore, this increase in *resolution distance* results in a decrease in *average distance* between the nodes. The anomalies in the results that

can be seen between 100 and 350 nodes deployed are due to the randomness in the deployment of nodes to the sensor field. As a result, in order to maintain higher efficiency in coverage, lower *resolution distance* (*r*) should be preferred.

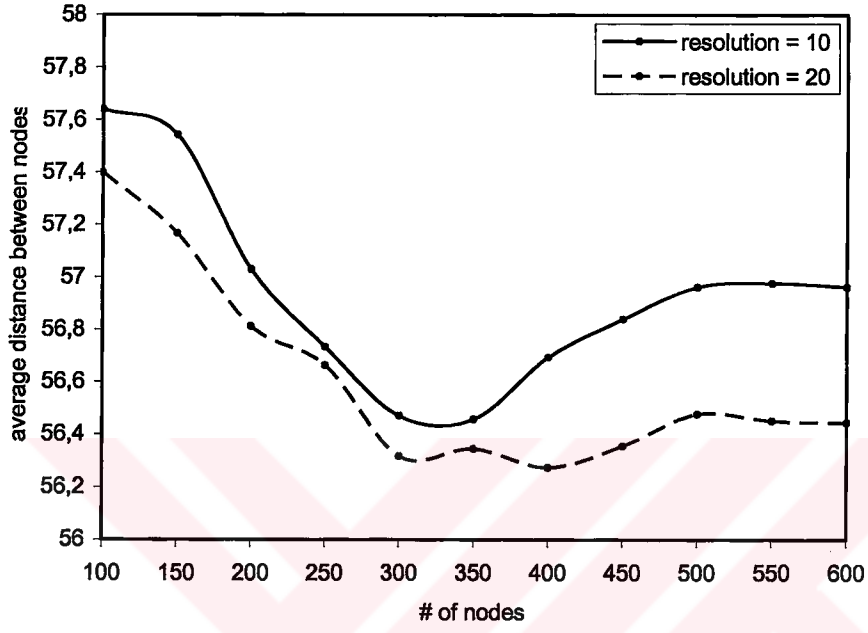



*Figure 23. Average distance ($\theta$) for varying resolution distances (r).*

## C. PERFORMANCE EVALUATION OF THE DISTRIBUTED 3D SPACE COVERAGE SCHEME FOR SENSOR NETWORKS CONSIST OF NON-LOCATION AWARE NODES

In this section, we evaluate the performance of the distributed scheme for the sensor networks consist of non-location aware nodes by the following experiments. We use the same values for the parameters explained in Section V.A during our simulations. Although the sensor nodes are lack of location information, in order to evaluate the performance of this scheme we separate the sensor space into unit cubes which has the edge length of 10 units *(resolution distance*) as we do in the former part.

### *1. Experiment-1: Sensitivity of Coordination Distance ($\alpha$) Over Coverage Efficiency ($\delta$)*

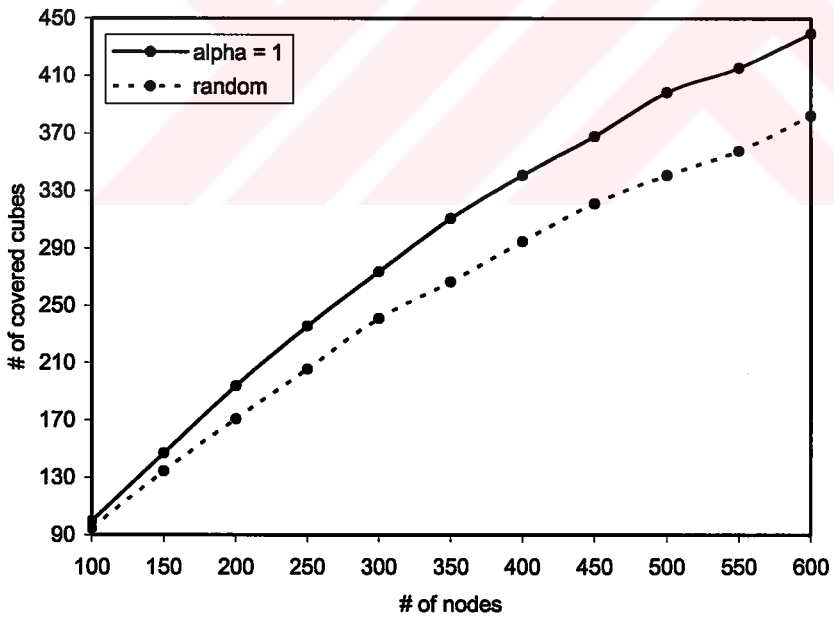



*Figure 24. Number of covered cubes with our scheme and random depth selection model.*

In Figure 24 we compare the number of covered cubes with this version of our scheme and random depth selection model for varying number of nodes deployed in the sensor field. As it can easily be seen in Figure 24 for every number of sensor nodes our algorithm covers more number of cubes and outperforms the random depth selection model.

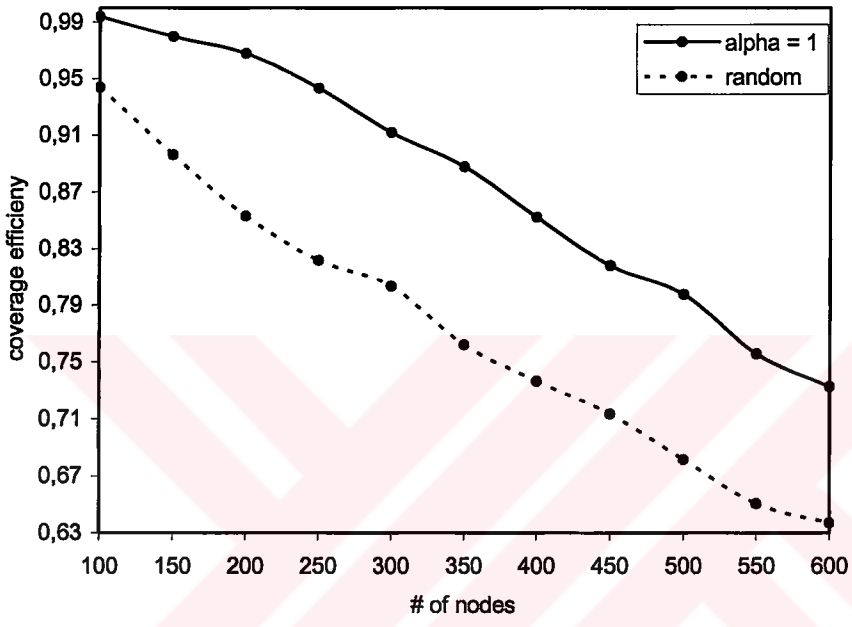



*Figure 25. Coverage efficiency ($\delta$).*

As we stated before, *coverage efficiency* ($\delta$) is the ratio of the number of covered cubes to the number of nodes in the sensor space. In Figure 25 *coverage efficiency* of our algorithm and the random depth selection model is compared for varying number of nodes deployed in the sensor field. Different from the previous scheme, in this algorithm nodes cover all the uncovered layers within the range of $\alpha$ hop neighboring nodes. Since the nodes are lack of location information they cannot determine in which column they are. Therefore, after all the layers are covered by $\alpha$ hop neighboring nodes they select the layer which is covered by the minimum number of nodes. Please note that nodes do not know in which cube they are, they only select a layer, we know the cubes of nodes in order to evaluate the performance of our algorithm. Although the calculated layer of a node is

covered by the minimum number of nodes there may be another node in the same cube of the layer. Therefore when the number of $\alpha$ hop neighbor nodes for a node exceeds the total number of layers there exist a possibility to be another node in the same cube of the layer. Unless the number of the nodes in the $\alpha$ hop range of a node overruns the total number of layers, there will be no reduction in *coverage efficiency* of our algorithm.

In Figure 26 and Figure 27 the number of cubes covered by our algorithm and *coverage efficiency* ($\delta$) of our algorithm is shown respectively for varying *coordination distances* ($\alpha$). When the *coordination distance* value increases, the $\alpha$ hop range of a node gets bigger and total number of neighbor nodes for a node rapidly exceeds total number of layers. Nodes exchange information with more neighbor nodes while they are calculating their depths. For that case a sensor node selects the minimum covered layer as its depth. Even when sensor nodes select a layer which is covered by minimum number of nodes, as the total number of sensor nodes increases, the probability of being another node in the same cube of the selected layer also increases. Therefore our algorithm produces more effective results for the *coverage efficiency* when lower *coordination distance* values are used.

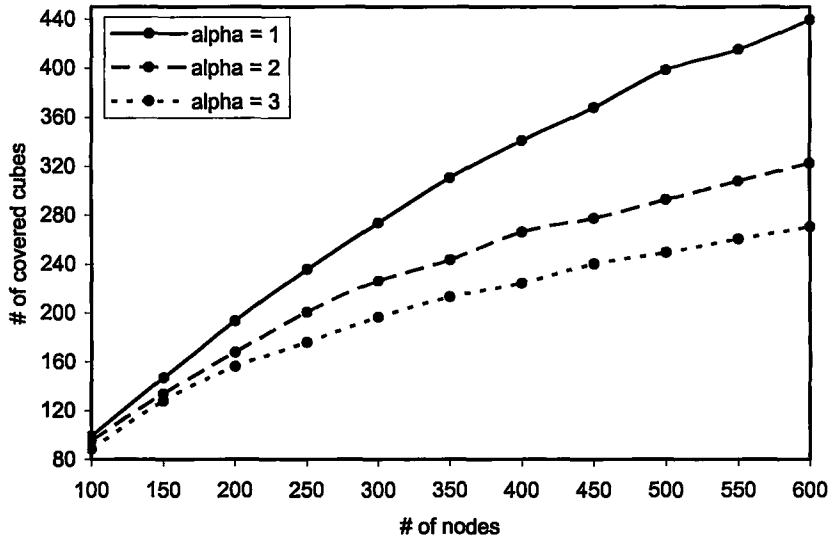



*Figure 26. Number of covered cubes for varying coordination distances ($\alpha$).*

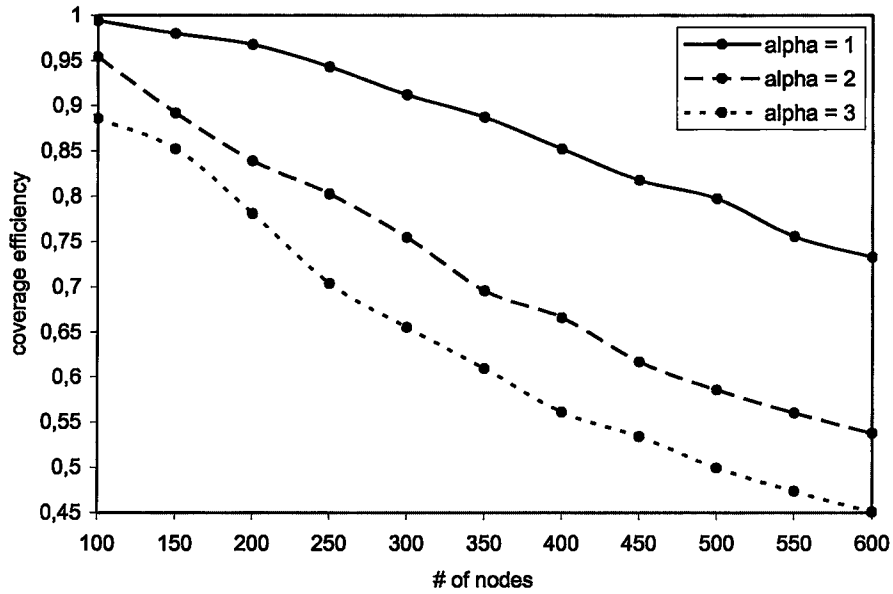



*Figure 27. Coverage efficiency ($\delta$) for varying coordination distances ($\alpha$).*

### *2. Experiment-2: Sensitivity of Coordination Distance ($\alpha$) Over Average Distance ($\theta$) Between The Nodes*

As we explained before the locations of nodes on the $x$ and $y$ axis (sea surface) have a significant importance on *average distances* between the nodes. Separate deployment of nodes on the $x$ and $y$ axis will make a positive effect on the increase of *average distance*. At the same time nearby deployment of nodes will make a negative effect on the increase of *average distance*. Therefore nodes should be deployed as much separate as possible to aid maximizing *average distance*. But in our simulations we deployed nodes randomly according to the uniform distribution and the performance of our scheme is evaluated based on random deployment of the nodes. Therefore we can only increase *average distance* by adjusting the depth of nodes in $z$ (depth) axis.

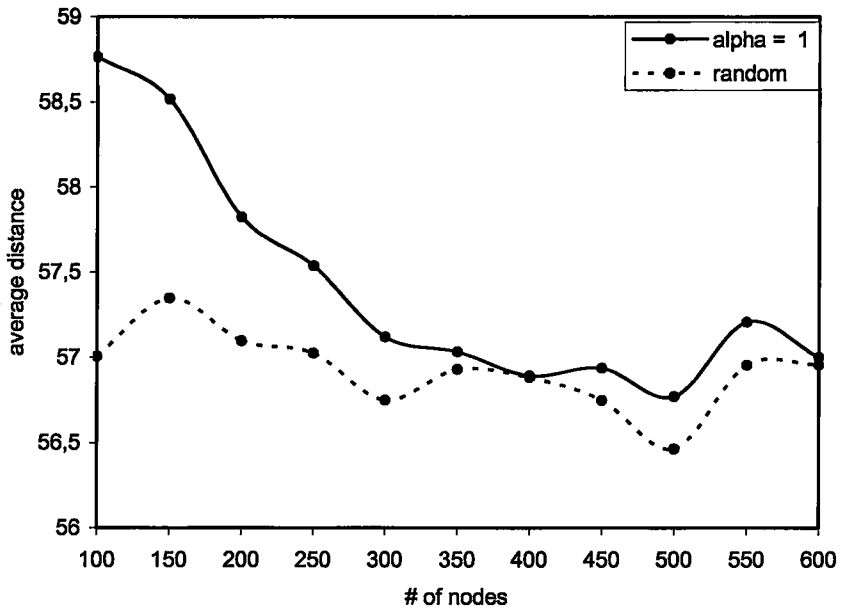



*Figure 28. Average distance ($\theta$).*

In Figure 28 and 29 *average distances* for our algorithm with varying *coordination distances* and random depth selection model are compared for various numbers of sensor nodes. Although the random depth selection model provides an even distribution of nodes in $z$ axis, as shown in Figure 28 our algorithm outperforms it. When the *coordination distance* increases the number of neighbors of a node also increases. According to our algorithm, when the total number of neighbors of a node overruns the total number of layers, node selects an uncovered layer if there is one. If there is not an uncovered layer node ignores maximizing *average distance* and selects the layer which is covered by minimum number of nodes in its $\alpha$ hop range. Therefore for the bigger values of *coordination distance* there will be a decrease in *average distance*. As it can easily be seen in Figure 29 after a certain number of nodes deployed, *average distance* is not sensitive to the changes in *coordination distance* and our algorithm produces the same results for all the *coordination distance* values as expected. The anomalies in the results are due to the randomness in the deployment of nodes to the sensor field.

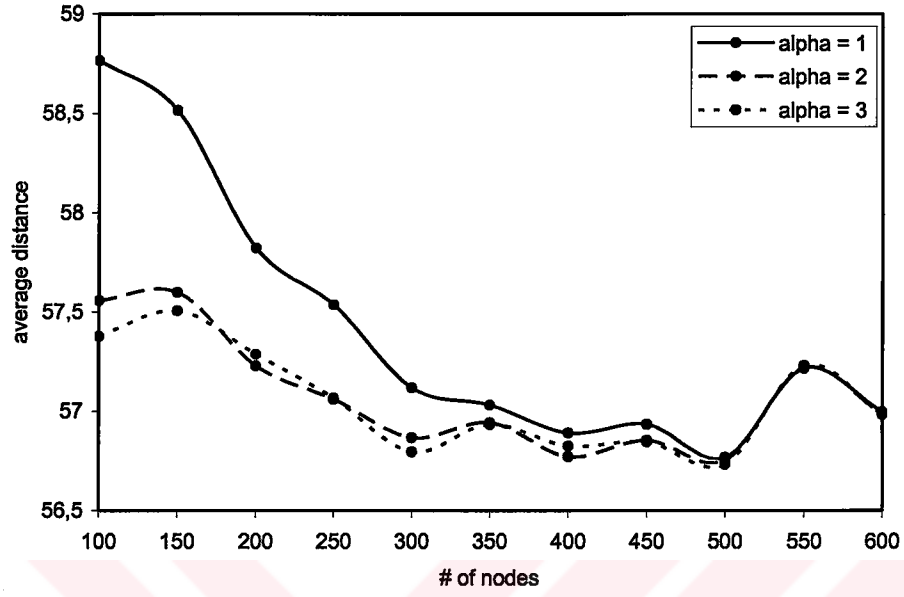



*Figure 29. Average distance ($\theta$) between nodes for varying coordination distances ($\alpha$).*

### *3. Experiment-3:Sensitivity of Coordination Distance ($\alpha$) Over The Average Number of Transmitted Control Packets Per Node*

As stated in Section III.B.1, the success of our algorithm depends on the control packets that are sent by each sensor node to be received by its every $\alpha$ hop neighbor nodes. In Figure 30 we compare the number of transmitted control packets per node for varying *coordination distances ($\alpha$)*. As we stated in Section III.B.2 nodes initialize the *ttl* field value of their *info* packets with the *coordination distance* ($\alpha$). Therefore increase in *coordination distance* means increase in the *ttl* field value of *info* packets. Bigger values of *ttl* field indicate that the packet is relayed more times than the lower values of *ttl* field values. In other words, increase in *coordination distance* results more neighbor nodes to exchange information. As the total number of sensor nodes increases the number of neighbor nodes of a node also increases. This means that more number of

neighbor nodes involve in forwarding of the *info* packet of a node. When *coordination distance* ( $\alpha$ ) is accepted as 1, nodes do not relay the *info* packets of their neighbors because the *ttl* field value of the packet is 1. Therefore as shown in Figure 30, except the dissemination of the *organize* packet, every node transmits only its own *info* packet when *coordination distance* ( $\alpha$ ) is 1. As we can see in Figure 30 the number of control packets sent by a sensor node increases for the higher *coordination distance.*

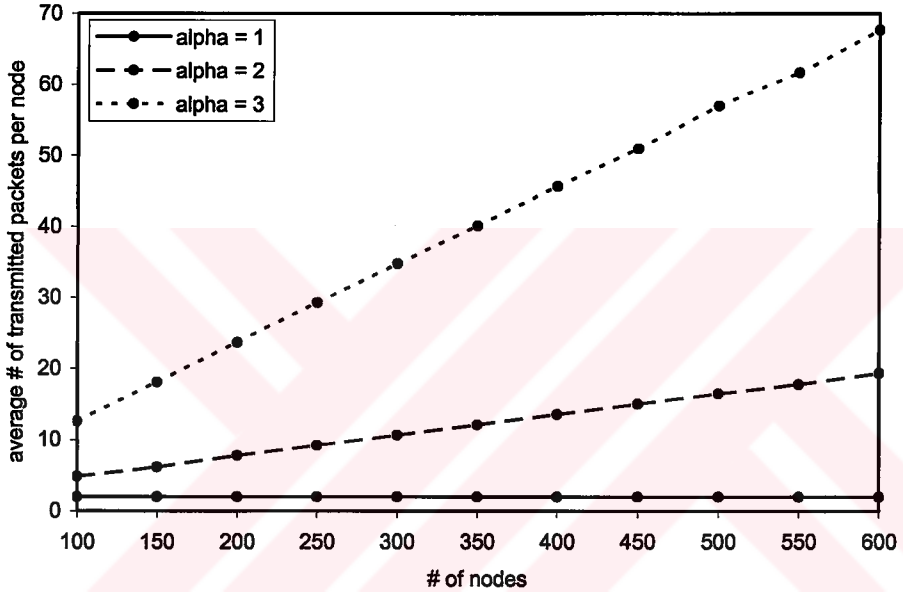



*Figure 30. Average number of transmitted packets per node for varying coordination distances ( $\alpha$ ).*

### *4. Experiment-4:Sensitivity of Coordination Distance ( $\alpha$ ) Over The Node Distribution to The Layers*

In Figure 31 sensor node distributions to the layers are shown for varying *coordination distances* ( $\alpha$ ) of our algorithm and random depth selection model when 600 nodes are deployed in our sensor field. Different from the previous scheme in this algorithm, nodes covers all the uncovered layers within the range of $\alpha$ hop neighboring nodes. After all the layers are covered nodes selects a layer which is covered by minimum number of sensor nodes. Since nodes negotiate

with more neighbor nodes for the bigger *coordination distance* ($\alpha$) values while they are evaluating their depths, they will have more precise information about the global node distributions to the layers. Thus they can make more accurate selection when they are determining their layer. As the *coordination distance* increase nodes are dispersed more homogeneous into the all layers.

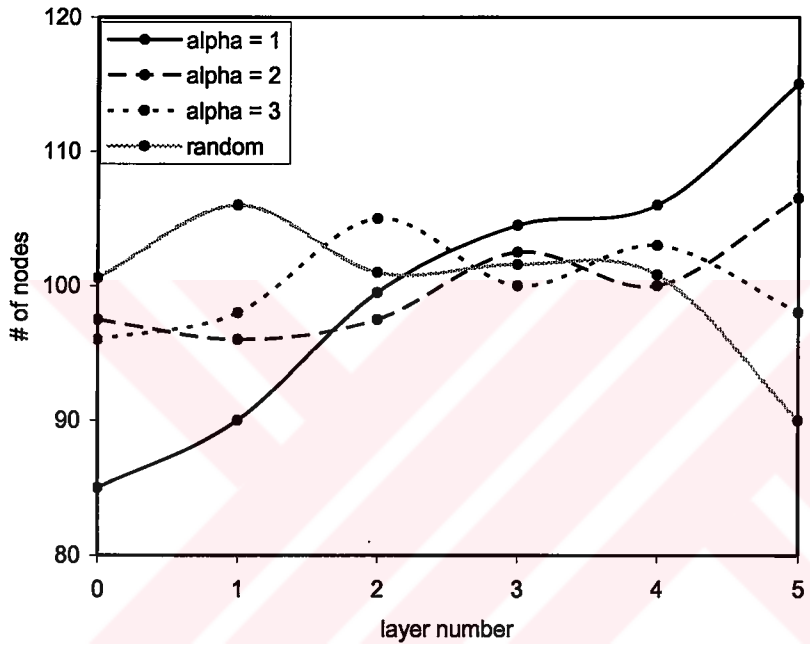



*Figure 31. The number of sensor nodes at each layer.*

### *5. Experiment-5: Sensitivity of Resolution Distance (r) Over Coverage Efficiency ($\delta$) and Average Distance ($\theta$) Between The Nodes*

As the *resolution distance (r)* value increases the dimensions of the unit cube that the sensor space is separated into also increases. When we partition our sensor space with the unit cube which has the edge length of 20 units (*resolution distance*) units we obtain 75 unit cubes and there will be only 3 layers. Therefore we use *node density* which is explained in Section IV.B.5 to compare the *coverage efficiency* of our algorithm with various *resolution distances*.

In Figure 32 coverage *efficiency* ($\delta$) of our algorithm for various *node densities* ($n_d$) are shown when *resolution distances (r)* is accepted as 10 and 20 units respectively. When bigger *resolution distance (r)* values are used, the size of the unit cube that the sensor space partitioned into also gets bigger. As a result of random deployment of nodes to the sensor field (sea surface), for the bigger *resolution distances* when the *node density* increases the number of nodes in the same cube exceeds the total number of layers. Different from the previous scheme, in this algorithm nodes cover all the uncovered layers within the range of $\alpha$ hop neighboring nodes. After all the layers are covered by $\alpha$ hop neighbors nodes select the layer which is covered by the minimum number of nodes. Although the calculated layer of a node is covered by the minimum number of nodes there may be another node in the same cube of the layer. Therefore when the number of $\alpha$ hop neighbor nodes for a node exceeds the total number of layers there exist a possibility to be another node in the same cube of the layer. Unless the number of the nodes in the $\alpha$ hop range of a node overruns the total number of layers, there will be no reduction in *coverage efficiency* of our algorithm.

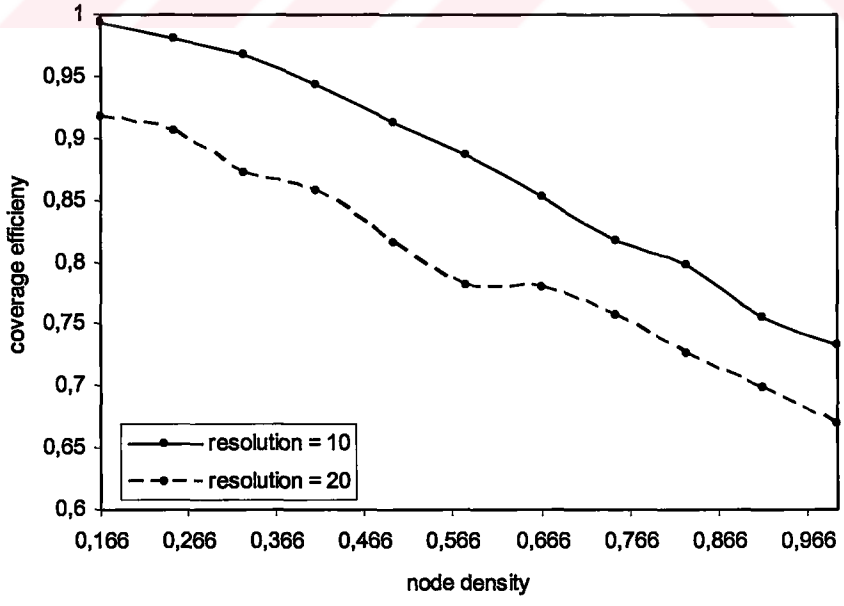



*Figure 32. Coverage efficiency ($\delta$) for varying node densities.*

In Figure 33 *coverage efficiency* ($\delta$) of our algorithm for various *resolution distances (r)* are compared when various number of nodes deployed to the sensor space. As we explain before, for the bigger *resolution distances* the dimensions of the unit cube increases. Therefore, for the higher *resolution distance* values, when the deployed sensor nodes increases the number of sensor nodes per a cube rapidly increases and this results in an obvious reduction in the *coverage efficiency*. In Figure 33 the dotted curve fits the *coverage efficiency* curve of our algorithm when *resolution distance* is 20 units and can be denoted as follows:

$$y = 0,699e^{-0,003x} \qquad (18)$$

where $y$ is *coverage efficiency and* $x$ is number of nodes in the sensor space.

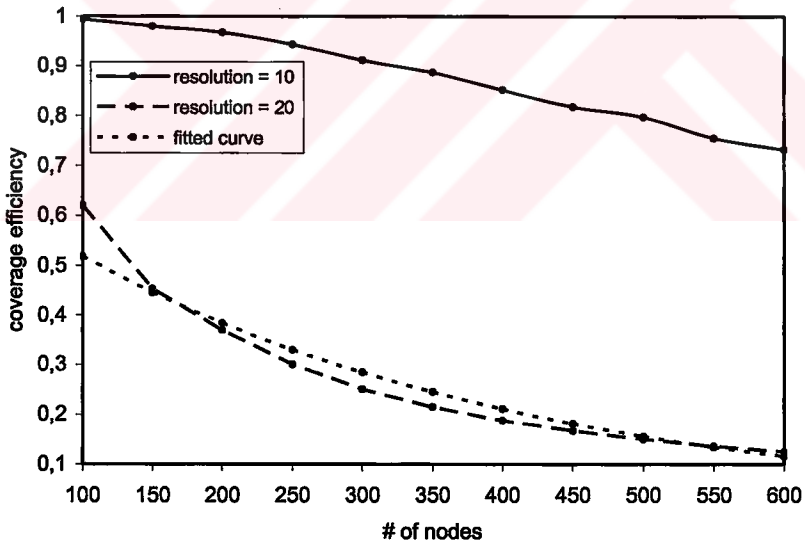



*Figure 33. Coverage efficiency ($\delta$) for varying resolution distances (r).*

In Figure 34 we compare *average distance* ($\theta$) of our algorithm for various *resolution distances* when *coordination distance* parameter accepted as 1. As we describe above, when *resolution distance* parameter is accepted as 20 units the volume of the unit cube increase and the total number of layers of the sensor

space decreases form 6 to 3.Therefore nodes are dispersed into 3 layers instead of 6. Since we have only 3 layers when the number of deployed sensor nodes increases the number of sensor nodes per a layer rapidly increases. Therefore, this increase in *resolution distance* results in a decrease in *average distance* between the nodes. The anomalies in the results are due to the randomness in the deployment of nodes to the sensor field. As a result, in order to maintain higher efficiency in coverage, lower *resolution distance* (*r*) should be preferred.

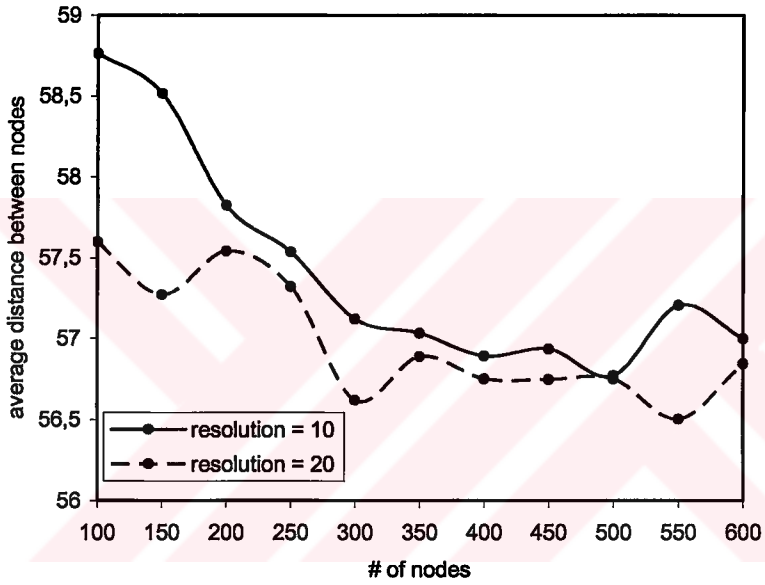



*Figure 34. Average distance ($\theta$) for varying resolution distances (r).*

### *6. Experiment-6: Sensitivity of Transmission Range of Nodes Over Coverage Efficiency ($\delta$), Average Distance ($\theta$) Between The Nodes and The Node Distribution to The Layers*

In Figure 35 *coverage efficiency* ($\delta$) of our algorithm is shown for varying transmission range of nodes when *coordination distance* ($\alpha$) is 1. When the transmission range of nodes gets bigger, the $\alpha$ hop range of a node gets bigger. Therefore nodes exchange information with more neighbor nodes while they are calculating their depths and total number of neighbor nodes for a node

rapidly overruns total number of layers. Then a node selects the minimum covered layer as its depth. Even when sensor nodes select a layer which is covered by minimum number of nodes, as the total number of sensor nodes increases, the probability of being another node in the same cube of the selected layer also increases. Therefore our algorithm produces more effective results for *coverage efficiency* when the radio module of nodes has lower transmission range.

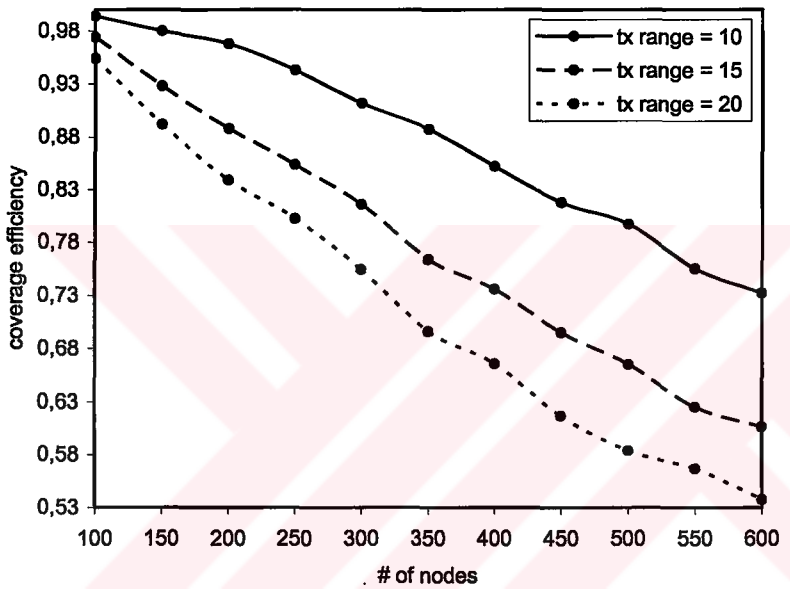



*Figure 35. Coverage efficiency ($\delta$) for varying transmission range of nodes.*

In Figure 36 we compare *average distances* ($\theta$) of our algorithm for varying transmission range of nodes when *coordination distance* ($\alpha$) is 1. As we explained above when the transmission range of nodes increases the number of neighbors of a node also increases. When the total number of neighbors of a node overruns the total number of layers node selects an uncovered layer if there is one. If there is not an uncovered layer node ignores maximizing *average distance* and selects the layer which is covered by minimum number of nodes in its $\alpha$ hop range. Therefore for the bigger values of transmission range of nodes there will be

a decrease in *average distance*. The anomalies in the results are due to the randomness in the deployment of nodes to the sensor field.

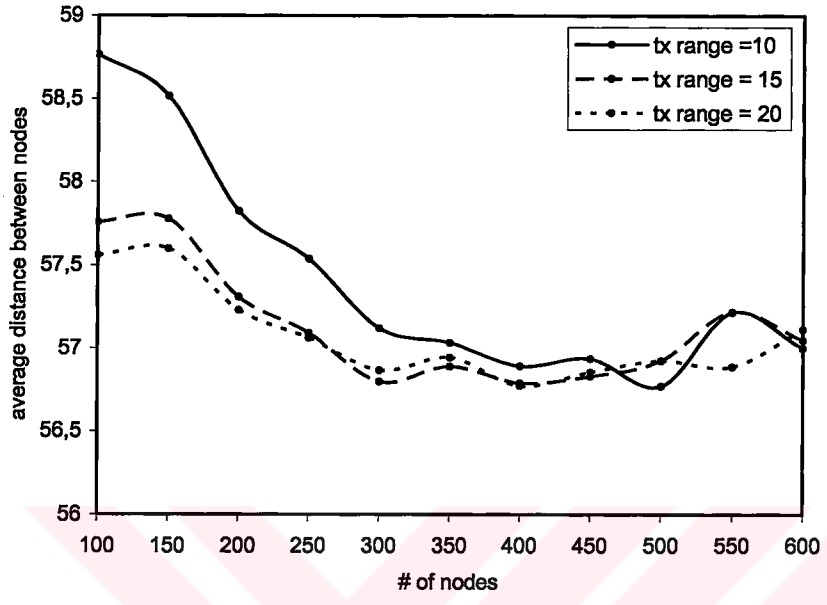



*Figure 36. Average distance ($\theta$) for varying transmission range of nodes.*

In Figure 37 node distributions to the layers are shown for varying transmission range of nodes when 600 nodes are deployed in our sensor field. As we stated before nodes negotiate with more neighbor nodes for the bigger transmission range of nodes while they are evaluating their depths, they will have more global information about the number of nodes at each layer. Thus they can make more accurate selection when they are calculating their layer. As the transmission range of nodes increases nodes are dispersed more homogeneous into all the layers.

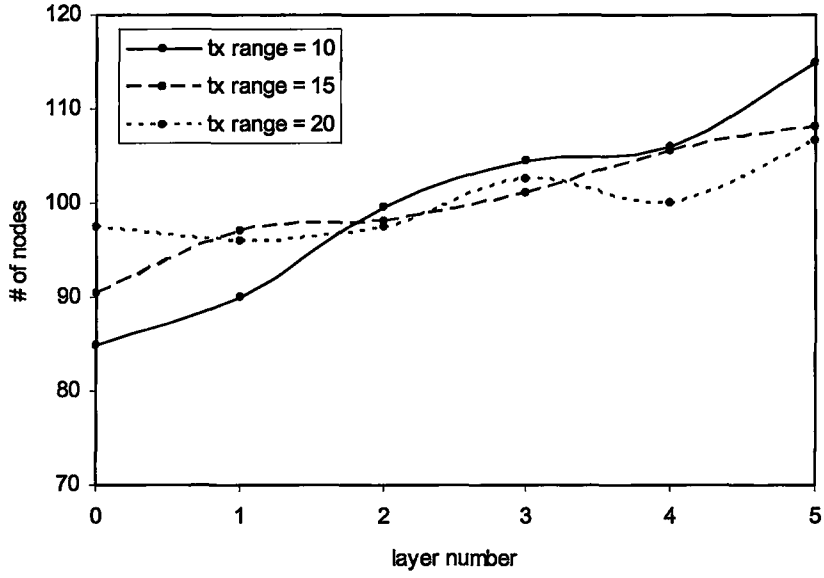



*Figure 37. The number of nodes at each layer for varying transmission range of nodes.*

# CONCLUSIONS

WSNs consist of a large number of ultra-small autonomous sensor nodes that are battery powered and equipped with integrated sensors, data processing capabilities, and short range radio communications [3][4]. In WSNs, sensor nodes gather data from their environment and convey these data to any end point using a wireless medium with a collaborated effort by using their limited power sources. Attractive features of these networks such as flexibility, low cost and self-organization let WSNs to find a place in many application areas.

We propose a novel architecture for underwater wireless sensor networks that can be used to detect a target in the vicinity of the sensor nodes. According to our architecture, when sensor nodes first deployed randomly, sensors lie in surface buoys. After deployment, nodes adjust their depths via a cable which is also used as the communication link between the sensor and the surface buoy. Although the sensors are underwater, the nodes can collaborate through the wireless medium over sea surface by using the antenna at the surface buoys.

We also develop two distributed placement algorithms in order to maximize the coverage of the 3D underwater sensor space of our proposed architecture. One of these algorithms is developed for sensor networks consist of location aware nodes and the other one is for the sensor networks consist of non-location aware nodes. Our distributed placement algorithms that rely solely on local packet exchanges run on sensor nodes and calculate the depths of sensors according to the locations of neighboring nodes such that the maximum 3D coverage of the sensor space is maintained. Although our schemes are distributed and adaptive, our experiments show that they maintain a high coverage of the sensor space in the expense of acceptable control traffic overhead.

As a future work, the impact of currents to the performance of our distributed schemes can be examined. Our distributed schemes can be improved to

counteract the currents that may be in different directions for different depths. We also experiment the applicability of the novel spatial query patterns based on Octtree addressing which are presented in Appendix 1 to query 3D underwater wireless sensor networks as further study.

# LIST OF REFERENCES


[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey", Computer Networks (Elsevier) Journal, pp. 393-422, March 2002.

[2] S. Tilak, N.B. Abu-Ghazaleh, W. Heinzelman, "A Taxonomy of Wireless Micro-Sensor Network Models", ACM SIGMOBILE Mobile Computing and Communications Review, April 2002.

[3] 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.

[4] G. Pottie, "Hierarchical Information Proccessing in Distributed Sensor Networks", Int'l Symp. Information Theory Conf. (ISIT.'98), pp. 163, August 1998.

[5] D. Estrin, R. Govidan, J. Heidemann, S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks", ACM/IEEE Conf. MobiCom'99, August 1999.

[6] J. Kahn, R.Katz and K.Pister, "Mobile Networking for Smart Dust", ACM/IEEE Conf. MobiCom'99, August 1999.

[7] C. Perkins, "Ad Hoc Networking", 1st Edition, Addison Wesley, 2001.

[8] D. Chen and P. K. Varshney, "QoS Support in Wireless Sensor Networks: A Survey", The 2004 International Conference on Wireless Networks (ICWN 2004), June 21-24, 2004.

[9] K. Sohrabi, J. Gao, V. Ailawadhi and G. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network", IEEE Personal Comm. Magazine, vol. 7, no: 5, pp.16-27, October 2000.

[10] R.W. Heinzelman, J. Kulik, and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Networks" ACM/IEEE Conf. MobiCom'99, August 1999.

[11] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", The 6th Annual International Conf. on Mobile Computing and Networking, pp. 56-67, August 2000.

[12] E.M. Sozer, M. Stojanovic and J. G. Proakis, "Underwater Acoustic Networks", IEEE Journal of Oceanic Engineering, Vol. 25, No.1, January 2000.

[13] S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Networks", IEEE INFOCOM'01, April 2001.

[14] B. Liuand, D. Towsley " On The Coverage and Detectability of Large Scale Wireless Sensor Networks", WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, March 2003.

[15] K. Kar, S. Banerjee "Node Placement for Connected Coverage in Sensor Networks", WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, March 2003.

[16] D.W. Gage, "Sensor Abstractions to Support Many-Robot Systems," SPIE Mobile Robots VII, vol. 1831, pp. 235-246, November 1992.

[17] A. Howard, M. J. Mataric, G. S. Sukhatme, "Mobile Sensor Network Deployment Using Potential Fields: A Distributed, Scalable Solution to The Area Coverage Problem", The 6th International Symposium on Distributed Autonomous Robotics Systems (DARS02), Fukuoka, JA, 2002

[18] S.S. Dhillon and K. Chakrabarty, "A Fault Tolerant Approach to Sensor Deployment in Distributed Sensor Networks", Army Science Conference, Paper ID: JP-05, 2002.

[19] S. S. Dhillon and K. Chakrabarty "Sensor Placement for Grid Coverage under Imprecise Detections", IEEE Transaction on Computers, vol.51, 2002.

[20] K. Chakrabarty, S. S. Iyengar, H. Qi and E. Cho "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks", IEEE Transaction on Computers, vol.51, pp.1448-1453, December 2002.

[21] Y. Zou and K. Chakrabarty, "Sensor Deployment and Target Localization Based on Virtual Forces", IEEE INFOCOM Conference, pp. 1293-1303, March 31-April 03, 2003.

[22] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan and K. K. Saluja, "Sensor Deployment Strategy for Target Detection", WSNA'02, 2002.

[23] K. Chakrabarty, S. S. Iyengar, H. Qi and E. Cho, "Coding Theory Framework for Target Location in Distributed Sensor Networks", International Conference on Information Technology: Coding and Computing (ITCC'01), pp. 130-134, April 01-04, 2001.

[24] T. Clouqueur, P. Ramanathan, K. K. Saluja, and K. Wang. "Value-Fusion versus Decision-Fusion for Fault-Tolerance in Collaborative Target

Detection in Sensor Networks", The Fourth International Conference on Information Fusion, August 2001.

[25] S. Poduri and G. S. Sukhatme, "Constrained Coverage for Mobile Sensor Networks", IEEE International Conference on Robotics and Automation, April 26–May 1, 2004.

[26] D. Tian and N. D. Georganas, "Connectivity Maintenance and Coverage Preservation in Wireless Sensor Networks", The 2004 IEEE Canadian Conference on Electrical and Computer Engineering, 2004.

[27] S. Slijepcevic, M. Potkonjak , "Power Efficient Organization of Wireless Sensor Networks", IEEE International Conference on Communications (ICC), pp. 472-476, June 2001.

[28] S. Meguerdichian, M. Potkonjak "Low Power 0/1 Coverage and Scheduling Techniques in Sensor Networks", UCLA Technical Reports 030001, January 2003.

[29] T. Yan, T. He and J. A. Stankovic "Differentiated Surveillance for Sensor Networks", The First ACM Conference on Embedded Networked Sensor Systems (Sensys03), November, 2003.

[30] S. S. Dhillon and K. Chakrabarty "Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks", Wireless Communications and Networking, WCNC 2003. IEEE, Vol:3, pp:1609 - 1614 ,16-20 March 2003.

[31] H. Gupta, S. R. Das and Q. Gu, "Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution", Mobihoc'03, Annapolis, MA, 2003.

[32] M. D. Berg, O. Schwarzkopf, M. V. Kreveld and M. Overmars, "Computational Geometry: Algorithms and Applications", 2nd Edition Springer-Verlag, 2000.

[33] M. Marengoni, B. A. Draper, A. Hanson and R.A. Sitaraman "System to Place Observers on a Polyhedral Terrain in Polynomial Time", Image and Vision Computing, pp.773-780 December 1996.

[34] A. Howard and M. J. Mataric, "Cover Me! A Self-Deployment Algorithm for Mobile Sensor Networks", International Conference on Robotics and Automations, Washington DC, May 2002.

[35] A. Howard, M. J. Mataric and G. S. Sukhatme "An Incremental Self-Deployment Algorithm for Mobile Sensor Networks", Autonomous Robots, Special Issue on Intelligent Embedded Systems, 13(2), pp. 113-126, September 2002.

[36] O. Khatib "Real-time Obstacle Avoidance for Manipulators and Mobile Robots", International Journal of Robotics Research, Vol. 5, No. 1, spring 1986.

[37] J. Elson and D. Estrin, "Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks", Int'l Conf. Distributed Computing Systems (ICDCS '01), April 2001.

[38] N. Bulusu, J. Heidemann and D. Estrin, "GPS-less Low-Cost Outdoor Localization for Very Small Devices", IEEE Personal Communication Magazine, vol. 7, pp. 28-34, October 2000.

[39] N. Bulusu, J. Heidemann and D. Estrin, "Adaptive Beacon Placement", Int. Conf. Distributed Computing Systems, pp. 489-498, 2001.

[40] J. Heidemann and N. Bulusu, "Using Geospatial Information in Sensor Networks", CSTB Workshop on Intersection of Geospatial Information and Information Technology, October 2001.

[41] R. Iyengar and B. Sikdar, "Scalable and Distributed GPS Free Positioning for Sensor Networks", IEEE ICC, International Conference on Communications, no: 1, pp. 338-342, May, 2003.

[42] K. Langendoen and N. Reijers, "Distributed Localization in Wireless Sensor Networks: A Quantitative Comparison", Computer Networks (Elsevier), vol.:43, pp.: 499-518, November, 2003.

[43] M. A. Weis, "Data Structures and Algorithm Analysis", 2nd Edition Addison-Wesley.

[44] K. K. Chintalapudi and R. Govindan, "Localized Edge Detection in Sensor Fields" , SNPA 2003.

[45] A. Erdogan, E. Cayirci and V. Coskun, "Sectoral Sweepers for Sensor Node Management and Location Estimation in Ad Hoc Sensor Networks", MILCOM 2003, Boston-USA.

[46] C. Cimen, E. Cayirci and V. Coskun, "Querying Sensor Fields by Using Quadtree Based Dynamic Clusters and Task Sets," MILCOM 2003, Boston-USA.

[47] E. Cayirci, "Data Aggregation and Dilution by Modulus Addressing in Wireless Sensor Networks", IEEE Communications Letters, August 2003.

[48] G. Kulkarni, C. Schurgers, M. B. Srivastava, "Dynamic Link Labels for Energy Efficient MAC Headers in Wireless Sensor Networks," IEEE International Conference on Sensors (Sensors'02), pp. 1520-1525, June, 2002.

# APPENDIX -1: OCTTREE ADDRESSING

Another challenge in WSNs is an effective addressing scheme needed to query nodes. The different characteristics of WSNs described in Section I.A necessities new addressing schemes where globally unique or fixed addressing usually is not a viable option for sensor nodes. Therefore we develop novel spatial query patterns based on octtree addressing to query 3D underwater wireless sensor networks effectively.

Addressing in WSNs can be classified into four categories;

1. **Attribute based naming and data centric routing:** Attribute based naming [8] is one of the earliest techniques used for node addressing in WSNs. In this technique, nodes that measure certain amplitude for a specified attribute are called, e.g. "nodes that measure more than $35^0$C temperature."

2. **Spatial addressing:** Spatial addressing is especially useful in applications such as intrusion detection and target tracking where queries are mainly based on node locations.

*a. Polygonal addressing:* In this technique the borders of a region is defined, and then the nodes inside, outside or in a buffer zone that has a certain depth along this border are queried. Borders of the region can be detected by a distributed edge detection [44] algorithm or can be specified by giving a series of geographical locations.

*b. Sectoral sweepers:* Spatial regions can also be specified by using a directional antenna, where nodes that receive a signal with a certain range of received signal strength indicator respond to a query [45].

*c. Quadtree based addressing*: Quadtrees can be used to partition the sensor field to quadrants, and then these quadrants are queried. Since a location aware node can easily find out which quadrant it is in for a given tree depth, nodes in the queried partitions respond. Various query patterns that can be created by this approach are examined in [46].

*d. Modulus addressing:* Hash functions can also be used to include or exclude nodes in queries [47].

***3. Using local identifications for nodes and mapping the destination of the user queries to the local identifications:*** In this scheme, every node is addressed by a local identification in the sensor field. Users may indicate a destination by using either an attribute based naming or spatial addressing scheme. A gateway node maps this address to a local identification.

***4. Address reuse:*** Address reuse is especially useful for MAC layer addressing of nodes. The same addresses can be assigned to multiple nodes as long as this does not cause a conflict. A distributed protocol for address reuse is proposed in [3]. Another approach for address reuse is dynamically assigning short link labels that are spatially reused to the links between the nodes instead of using conventional MAC address [48].

In our addressing technique, we represent the sensor space with a 3D spatial data structure, octtree. This space is accepted as the root of an octtree. At the first step, we divide this volume into eight equal sub volumes. After this process every sub volume is thought as a new parent, and divided into eight equal sub volumes until there will be only one sensor node in every sub volume. The leaf volumes are the final sub volumes of the octtree.

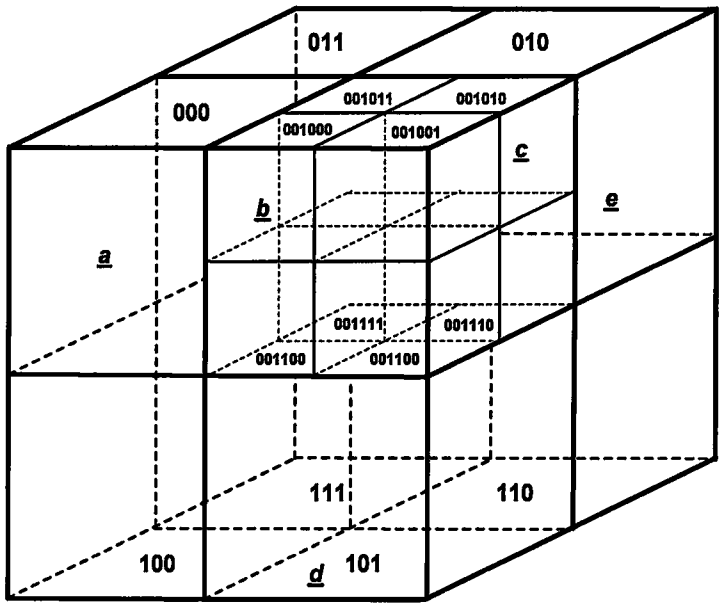



*Figure 38. The octtree partitioning of a sensor space.*

For any parent of octtree there exist eight children, and those can be addressed as; 000, 001, 010, 011, 100, 101, 110, 111 respectively. The whole sensor space can be addressed via our octtree approach by dividing the volume into as many sub volumes as required. Hence every sensor node belongs to a unique sub volume we can address each sensor node by the address of this sub volumes.

A sensor network composed of 5 sensor nodes and its octtree is illustrated in Figure 38 and 39. As an example, the octtree addresses of the sensor nodes in Figure 38; the address of a: 000, the address of b: 001000, the address of c: 001010, the address of d: 101, the address of e: 010. The addressing bit string is not fixed, and varies according to the location of sensor node. Each level of octtree, addressing length increases 3 bits. For a level $l$ we can address $8^l$ nodes. Thus $8^{n/3}$ nodes can be addressed by using $n$ bits.

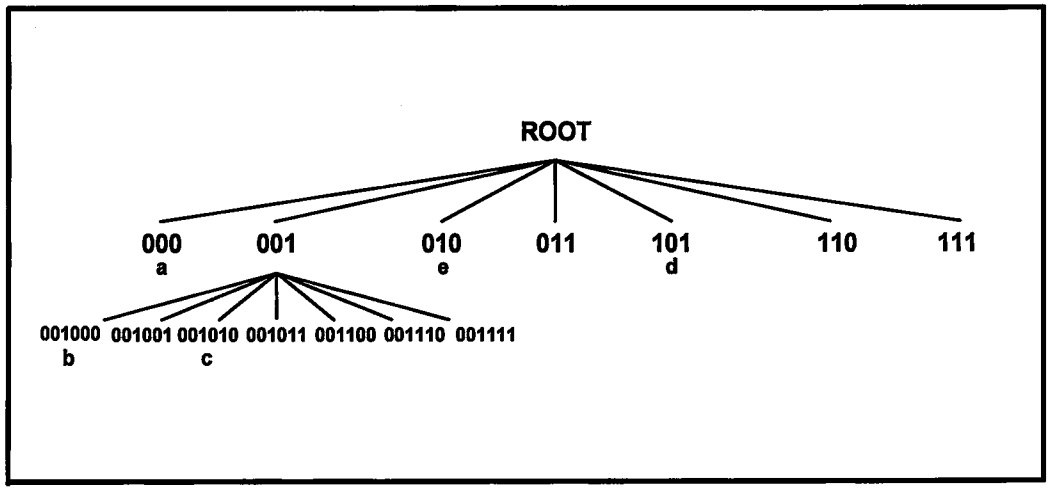



*Figure 39. The octtree of the example sensor field.*

Octtree addressing is very feasible for geocasting. For example the address "001" both addresses nodes b and c. Therefore 1/8 of the sensor field can be queried by using only 3 bits. More complex query patterns can also be designed by using octtree addressing as shown below:

*Pattern1* can be used when we know the expected depth interval of an intruder. Or (+) function is used to generate more complex octtree address. The shaded volumes in Figure 40 can be addressed by using the following octtree address: **0001** + 0011** + 0101** + 0111****.

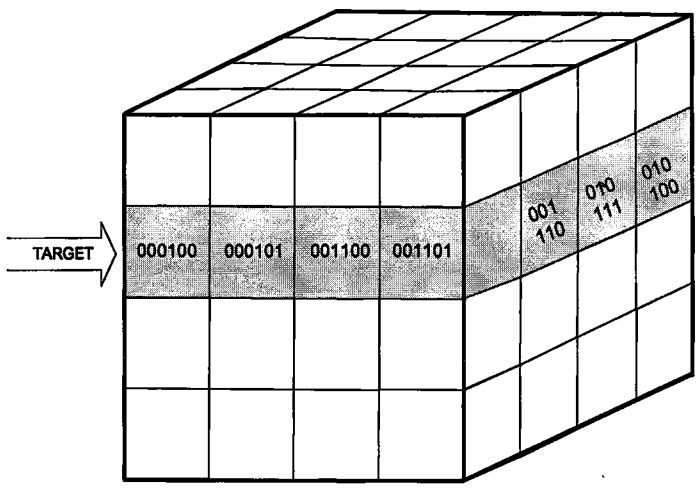



*Figure 40. Octtants queried by pattern 1.*

*Pattern 2* can be used when we know the expected size of the target. For example if the expected target covers a space of $t \times t \times t$ unit dimension in a sensor space , it is sufficient to partition the sensor space into $k$ sub volumes of $d \times d \times d$ , i.e, $d < t$, unit dimensions and to query $k/2$ sub volumes instead of querying the whole sensor space. This pattern is illustrated in Figure 41 and can be denoted by using the following octtree address: *****000 + ***010 + ***101 + ***111.**

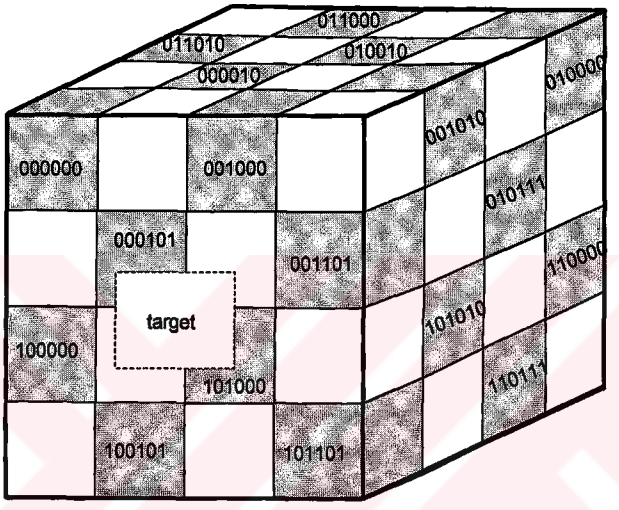



*Figure 41. Octtants queried by pattern 2.*

*Pattern 3* queries a border zone of the sensor space. Figure 42 is an example for this pattern and can be addressed as **000*0* + 001*0* + 100*0* + 101*0*.**

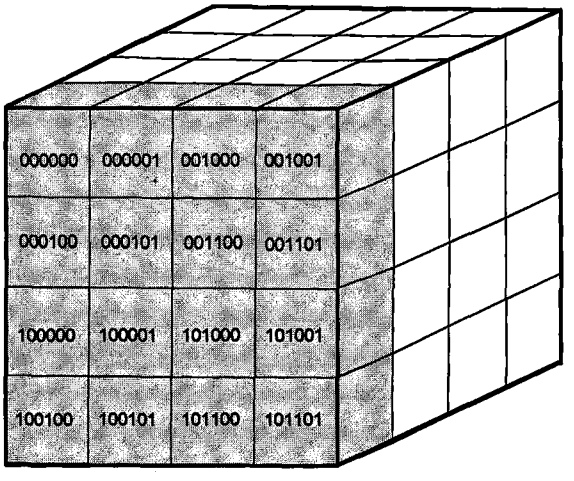



*Figure 42.Octtants queried by pattern 3.*

Users of the system use the octtree generated by the sink or an external proxy to create their queries. When a sensor node receives a query it checks whether it involves in the query. Nodes do not have to process the whole address string. They process the address string by three bits and keep on reading as long as the previous bits point their octtree region. Otherwise, sensor nodes understand that they are not expected to reply the query and discard it.

Octtree addressing necessitates location awareness of nodes. Especially the applications such as intrusion detection and target tracking sensed data should also be associated with location information. Practical techniques to find out node locations presented in [38][39][40][41][42]. We consider that octtree addressing can be used for the architecture we develop for the underwater wireless sensor networks. Since nodes adjust their depths to desired depths, exact 3D locations of the sensor nodes can easily be found by appending this depth ($z$) value to the $x$ and $y$ locations of the surface buoys. We improve our octtree based query patterns and experiment applicability of this new spatial addressing scheme as further study.