Unmanned Underwater Exploration Based on Spider Web Based Algorithm

Mrs.J.Maheswari *, S.Mohideen Hizana[*2], , N.Madhubala [*3] M.Saraswathi*,R.Soniya*
Department of Electronics and Communication Engineering & P.S.R.Rengasamy college of engineering for women,Virudhunagar,Tamilnadu,India

Abstract
Underwater sensor networks(UWSNs) face specific challenges due to the transmission properties in the underwater environment.Radio wave propagate only for short distances under water,and acoustic transmission have limited data rate and relatively high latency.One of the possible solutions to these challenges involes the use of autonomous underwater vehicles(AUVs) to visit and offload data from the individual sensor node.We propose spider web based path planning model to collect data from sensor nodes.in order to improve efficiency and reduced energy consumption AUV follows intersection node path from source to destination NS2tool has been used to evaluate existing and proposed system performance.

Keywords - NS2 simulator,AUV,LSNR,Sensor node

I. INTRODUCTION

A Underwater Sensor Network is a network of tiny sensor nodes which communicate with each other through a underwater communication link. Each sensor node typically consists of a processing device, small amount of memory, battery and radio transceiver for communication. These sensor nodes obtain data, e.g. temperature, pressure and humidity, do some local processing, and transmit the data to a neighbor node or a beacon node, which, in turn, may be connected to a central computer where major processing is performed. As is evident, this central computer may be part of a bigger computer network so that the information can be communicated from this central computer to other computers which are part of the
bigger network. The underwater sensor networks can be used in diverse applications in both industrial and commercial environments. Some of the most commons applications of underwater sensor networks include object tracking, habitat monitoring, fire detection, traffic monitoring and area monitoring. Some of the typical characteristics of sensor networks are small sized nodes, mobile nodes, a dynamic network topology, harsh operating environments and limited energy or power resource that these nodes should utilize efficiently as these may remain in an area for years without more energy being available.

There are many challenges involved at various levels and stages in the development of underwater sensor networks as discussed by Akyildiz et al. For example, the physical layer design of a underwater sensor node, which should be very small and yet accommodate all the functions that it is required to perform, presents many challenges. Similarly, new algorithms and protocols from link layer to application layer need to be developed. New operating systems to run on such tiny nodes are needed and to write these tiny operating systems, there is a need to develop new programming languages and new programming paradigms. One of the important problems is the path planning of sensor nodes i.e. determination of positions of nodes in the sensor field. This is important due to various reasons. For example, the data collected by a sensor node must be ascribed to the location from where it was collected. The data would not be useful if the location where it belongs to is not known. The set of values of temperature and humidity, for instance, collected by the sensor nodes are not meaningful unless the respective position coordinates from where these values were recorded are known.

An example of underwater sensor network application where location information is important is target tracking. Likewise, in a sensor network meant for earthquake disaster relief, the sensor positions must be known to ascertain the location of survivors buried somewhere in the rubble of a collapsed building. Similarly, one of the biggest challenges in sensor networks is the efficient utilization of energy resource which is not easily available to sensor nodes. And one of the most energy dependent operations is data transmission from sensor nodes to base stations which should use some energy-efficient and energy-aware routing algorithm. One of the approaches being worked out and which holds great promise is geographic location based routing, which is based upon mathematical modeling of sensor positions instead of using IDs. Again for location-based approach to be possible, the locations of the nodes must be known.

Due to various constraints, existing path planning systems, such as GPS, cannot be used for the path planning of underwater sensor nodes. Therefore, new strategies and algorithms for the path planning of sensor nodes are needed to be designed

and developed. The algorithms should be designed within the constraints defined, the characteristics of the sensor nodes and sensor network. In underwater sensor networks using only static sensor nodes, path planning algorithm usually runs only at the time of initial deployment of the nodes.

However, in a sensor network using mobile sensor nodes, the path planning algorithm is needed to determine the new positions of mobile nodes as they move in the sensor field. Hence, path planning algorithms for mobile sensor nodes need more energy compared to algorithms designed for static sensor nodes.

II. BACKROUND

Certain applications of underwater sensor networks require that the sensor nodes should be aware of their position relative to the sensor network. For it to be significant and to be of value, the data such as temperature, humidity and pressure gathered by sensor nodes must be ascribed to the relative position from where it was collected. For this to happen, the sensor nodes must be aware of their positions. The literature has come to term this problem of location or position estimation of sensor nodes simply as *localization*.

The term path planning has earlier been used in robotics where it is used to refer to determination of location of a mobile robot in some coordinate system. Under certain circumstances, the nodes should not only by aware of their position but also the direction or orientation relative to the network.

In a sensor network, the nodes may be categorized as:
Dumb Node (D)

It is the node that does not know its position and which would eventually find its location and position from the output of the path planning algorithm under investigation. Dumb nodes are also known as *free* or *unknown* nodes.

Settled Node (S)

A settled node is a node which was initially a dumb node but managed to find its position using the path planning algorithm.

Beacon Node (B)

A beacon node is a node that knows its position from the very start and always knows its position afterwards also without the use of path planning algorithm. It has a mechanism other than the path planning algorithm to find its position. For example, the beacon node may be equipped with a GPS device or it may be placed at a position with known coordinates. The beacon nodes are also called *reference nodes*, *anchor nodes* or *landmark nodes*.

It should be noted that sensor nodes may have *symmetric* or *asymmetric* communication links. If two nodes *u* and *v* are symmetric then *u* reaches *v* and *v* reaches *u* as well. In the case of asymmetric communication links, either *u* reaches *v* or *v* reaches *u* but both *u* and *v* do not reach each other simultaneously.

Let us now consider a sensor network which is symmetric, two-dimensional and arranged in a square shape. Then this sensor network can be represented as a graph *G(V, E)* where the set of sensor nodes can be represented as set of vertices as under:

$$V = \{V_1, V_2, \ldots, V_n\}$$

The set of edges *E* in the graph *G(V, E)* comprises of all edges $e = (i, j) \pounds E$ iff $v_i$reaches v_ji.e. the distance between $v_i$and $v_j$is less than *r* where *r* is the maximum distance between the two nodes after which communication between them ceases to exist i.e. if the distance between two nodes is greater than *r*, no direct communication between them is possible. In other words, if the distance between two nodes is greater than *r*, the two nodes are not *neighbor* nodes. The distance between two neighbor nodes *vi* and *vj* is defined as the weight $w(e) \leq r$ of the edge $e = (i, j)$ between them.

It is to be noted that problem of path planning is usually solved only for two dimensions with the supposition that when needed or deployed, it could be extended to three dimensions. It is for this reason, we have stated graph *G(V, E)* to be two-dimensional. Therefore, it can be stated that *G* is a Euclidean graph in which every sensor node has a coordinate $(xi, yi) \pounds R^2$ in a two-dimensional space. The coordinate *(xi, yi)* represents the location of a node *i* in the given sensor field.

CLASSIFICATION OF PATH PLANNING ALGORITHM: Majority of the existing path planning algorithms may be classified as *ranged-based* or *range-free* depending upon whether the algorithm uses distance estimation or some other information for estimating the node locations. Range-based algorithms usually use sensor field geometry information to determine node
locations. Communication between beacon nodes and dumb nodes also helps determine this geometric information about their relative placement e.g. distance between the two nodes or the angles of a triangle formed by the beacon nodes. This information is then further used to determine node location. When distance is used as a primary means to determine node location, this is termed as *lateration*, and when angle information is used for localization, it is known as *angulation*. For node path planning in a plane, precise distance measurements from at least three beacon nodes are required and we use *trilateration* for position estimation of a node.

Intersection of three circles around the three beacon nodes gives a single point as position of the node as is shown in Figure.

The same technique can be extended to three-dimensional space by the addition of a fourth beacon node. However, in actual practice, distance measurements are seldom precise and intersection of three circles may result in more than one point. The scheme may be improved by employing more than three beacon nodes for a plane and we then use *multilateration* to calculate the node position. In the *angulation* technique, angle information is used to deduce position of the node. Two beacon nodes and a dumb node form the vertices of a triangle and the lines joining them form the sides of the triangle. As the positions of beacon nodes are known, the distance between them, that is, one side of the triangle is known. If the two angles that the dumb node forms with the two beacon nodes are measured, the location of the dumb node can be calculated as third point of the triangle. This method of determining location of a node is termed as *triangulation*

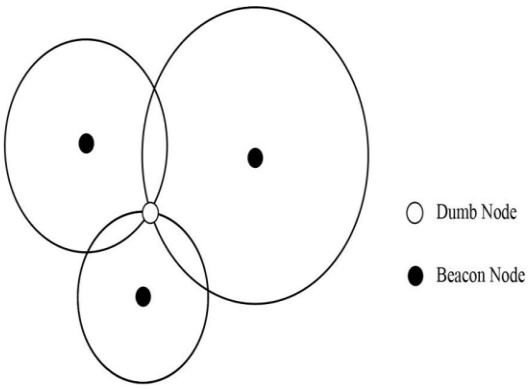

Fig.1.1 Trilateration

Intersection of three circles around three beacon nodes gives position of the dumb node

In *range-free* path planning algorithms, a node determines its position merely by finding the beacon nodes in its proximity and for this reason, range-free algorithms are also termed as *proximity-based* or *connectivity-based* algorithms. Such algorithms usually provide coarse-grained localization. However, with sufficient number of beacon nodes with overlapping transmission regions, a more accurate path planning is possible. Range-free algorithms are robust against the fluctuations of the underwater channel as the decision whether a node is in proximity of another node is based upon connectivity information sampled over a long period of time. Hence, short and temporary variations in the underwater channel do not affect the accuracy of location estimation. A range-free algorithm calculates the position without having to find the distance between the sensor nodes. A range-based path planning algorithm is more accurate but has major computational cost and usually additional hardware and hence increased energy requirements. On the other hand, a range-free algorithm is less accurate but does not require additional hardware and has smaller computational overhead.

Path planning algorithms may also be classified either as *centralized* or *distributed* depending upon whether they use central processor to estimate the positions of all nodes or else nodes performs local processing to determine their positions. In the case of centralized approach, node positions are calculated at a central processor. To calculate positions of distant nodes, the central unit usually needs certain parameters from these nodes, which depend on the particular path planning algorithm. These parameters from the distant nodes are sent to the central unit using the transceiver unit. The central unit computes positions of all nodes and sends the results back to them. As is evident, this approach may involve a lot of communication overhead for the dumb nodes, which, in turn, would require these nodes to have more battery power for sending and receiving the parameters. The centralized approach also introduces a single point of failure. If the central unit fails for some reason, the entire path planning process failsk. This approach is also prone to additional delays due to propagation delays involved in the transmission of parameters and results and the large amount of processing involved at the central unit. Furthermore, central processing approach is not suited to the very nature of ad-hoc networks.

In decentralized or distributed approach, the dumb nodes determine their position themselves by performing local processing. The dumb nodes usually need information from neighbor nodes and beacon nodes to be able to determine their position. For decentralized approach to work, the dumb nodes must possess local processing capability. However, communication overhead is less compared to the centralized approach.

Path planning algorithms may also be classified as *fine-grained* or *coarse-grained*. Some applications of sensor networks need to determine only symbolic location of a sensor node e.g. whether sensor node is inside room A or room B. This type of symbolic estimation of node position is termed as coarse-grained localization. In some other applications, we need to determine a more accurate estimate of node position e.g. node X is located at coordinates (x, y). This type of node path planning is termed as fine-grained localization.

Another classification of the path planning algorithms depends upon whether the algorithm is designed for *outdoor* unconstrained environment or for the *indoor* constrained environment. An earliest

path planning algorithm for the outdoor environment is due to Bulusu, Heidmann & Estrin. Yet another way to classify path planning algorithms is to consider whether the algorithm is designed for a sensor field in which the sensor nodes are *fixed* or for a sensor field comprising of *mobile* sensor nodes.

Majority applications of underwater sensor networks use static nodes. As a result, many of the path planning algorithms consider sensor networks comprising entirely of static nodes only. However, a few applications of underwater sensor networks deploy mobile nodes and hence a few path planning algorithms, such as the one by Kim & Kim, take the mobile nature of the nodes into consideration.

CHARACTERISTICS OF PATH PLANNING ALGORITHM: The main objective of a path planning algorithm is to determine position of a node. However, there are certain criteria that the algorithm should meet for it to be practicable. The criteria usually depend upon the type of application for which the path planning algorithm is designed. General design objectives or desired characteristics of an ideal path planning algorithm are:

- ✓ It is highly desirable that the path planning algorithms are *RF-based*. The sensor nodes are equipped with a short-range RF transmitter. An efficient path planning algorithm exploits this radio capability for path planning in addition to its primary role of data communication.
- ✓ A underwater sensor network is ad hoc in nature. The path planning algorithm should take the *ad hoc* nature of the network into consideration.
- ✓ The nodes should be able to determine their position in as small time as possible so that the path planning algorithm has a *low response time*. This would enable sensor nodes to be deployed quickly.
- ✓ The position of the sensor node found by such an algorithm should be *accurate* enough for the specific application for which this algorithm is being used.
- ✓ The algorithm must be *robust* so that it may work in adverse conditions.
- ✓ The algorithm should be *scalable* so that if sensor nodes are added or removed, it should still be able to work out the position of the nodes. Furthermore, the algorithm should produce acceptable results for sensor networks comprising of small to large number of nodes.
- ✓ The path planning algorithm should be *energy efficient* and preferably *energy aware* as well because the sensor nodes are autonomous and normally do not have any external source of power.
- ✓ The path planning algorithm should be *adaptive* to the change in the number of beacon nodes. If the number of available beacon nodes changes, the algorithm should still be able to provide location estimates. However, the accuracy of node estimates will change with the change in number of available beacon nodes. In general, with a higher number of beacon nodes, a path planning algorithm is able to compute more accurate estimates of node positions.
- ✓ The algorithm should be *efficient* so that it is able to compute node locations with as small number of beacon nodes as possible.
- ✓ The algorithm should be *universal* so that it is able to compute node locations under all conditions of changing environments and weather. In particular, it should work in constrained environments such as indoors and unconstrained environments such as outdoors.

Only an *ideal* path planning algorithm will be able to meet all the goals stated above. The path planning algorithms in practice will only meet a subset of these characteristics depending upon the particular application for which it is designed.

OPEN: Optimized Path Planning Algorithm with Energy Efficiency and Extending Network − Lifetime in WSN:

Syed Bilal Hussain et al proposes a new distributed clustering protocol for both homogeneous and heterogeneous environments, named Optimized path planning algorithm with energy efficiency and extending network lifetime in WSN (OPEN). In the proposed protocol, we use timer value concept for efficient CH selection based on multiple parameters, e.g., residual energy, the average distance from its neighbors, and node density.

III. EXISTING METHOD

Underwater sensor network is a collection of small devices called sensors nodes, which are deployed in the sensing field to monitor physical and environmental information. Location information of sensor node is a critical issue for many applications in underwater sensor network. The main problem is to design a path for a mobile landmark to maximize the location accuracy as well as to reduce energy consumption. Different path planning schemes have been proposed for localization. Here, this study focused only on static path planning scheme. In this article, the performance of five static path planning schemes is evaluated, namely, random way point, Scan, D-Scan, Hilbert, and Circles based on three parameters such as location error ratio, energy consumption, and number of references. Network simulator-2 is used as a simulation tool. Simulation scenarios with three node densities are used in this research study such as sparse node density, medium node density, and dense node density.

A. Random Way Point (RWP)

RWP mobility model is widely used in research areas because of its simplicity. The issue in random mobility model is that it cannot cover the whole ROI to localize all unknown nodes as shown in Figure. Here, each point is traversed many times by the mobile beacons while some points never visited. Another issue is that the path length travels by the mobile landmark is not possible to formulate in RWP. After a specific time, the movement would be terminated. The collinearity issues are expected to be minimum in RWP due to random movement of mobile beacon node.

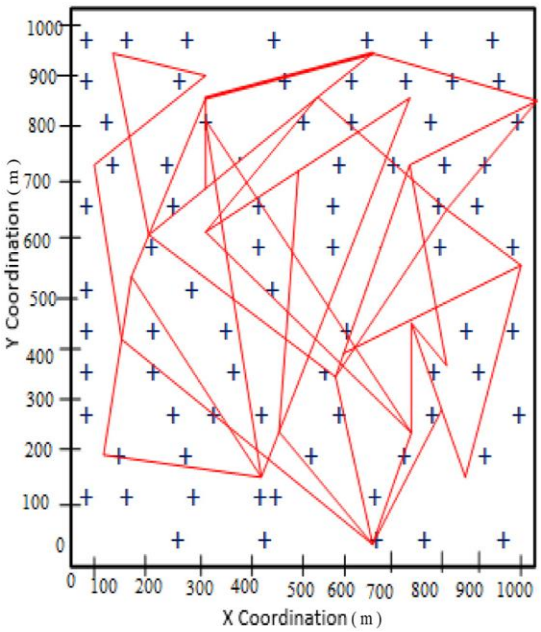

Fig.3.1 Random way point trajectory of mobile beacon.

In random waypoint mobility model, nodes are free to move randomly anywhere in the simulation field independent of each other. No restriction is imposed on them. There is random selection of destination, speed and direction with inclusion of pause times between changes in direction and/or speed. After being at a location for certain duration of time, mobile node chooses random destination and speed at pre-defined range and proceeds towards newly chosen destination with velocity chosen. On reaching destination, mobile node pauses for a fixed time period before restarting the same process again. Here, speed and pause time help in defining the mobility behavior of nodes. Low speed and long pause time result in stable network topology whereas high speed and short pause time leads to dynamic topology.

B. Scan trajectory

The scan is a simple path planning scheme. The scan can be implemented easily. The traveling trajectory of Scan is shortest. Here, the mobile anchor moves along a single dimension either along x-axis or y-axis as mentioned in Figure. One of the advantages of Scan is that it covers the whole network and has lowest path planning error. However, Scan has the collinearity issue which mostly occurs due to the movement of mobile beacon in a straight line, which degrades the path planning accuracy. Here, the network is divided into nxn sub-squares. The distance between two parallel lines to the Y-axis is defined as resolution denoted by d. The resolution should be small to make sure that all sensors will be able to receive the beacon messages. The resolution should not be more than d=2r, where r denotes the communication range. Scan performs best when the small. The total distance (Dscan) traversed by the mobile landmark in Scan is calculated by

$$D_{scan} = (4^n - 1)R$$

C. Double Scan trajectory

To overcome the collinearity issues, Double Scan traverses the network region along both directions x-axis and y-axis as shown in Figure. The collinearity problem is resolved in Double Scan. Double Scan also has a drawback. Here, the path length is doubled; due to this, the energy consumption is also increased. The total distance traversed by the mobile landmark in Double Scan is calculated by

$$D_{Doublescan} = (4^n + 2^n - 4)R$$

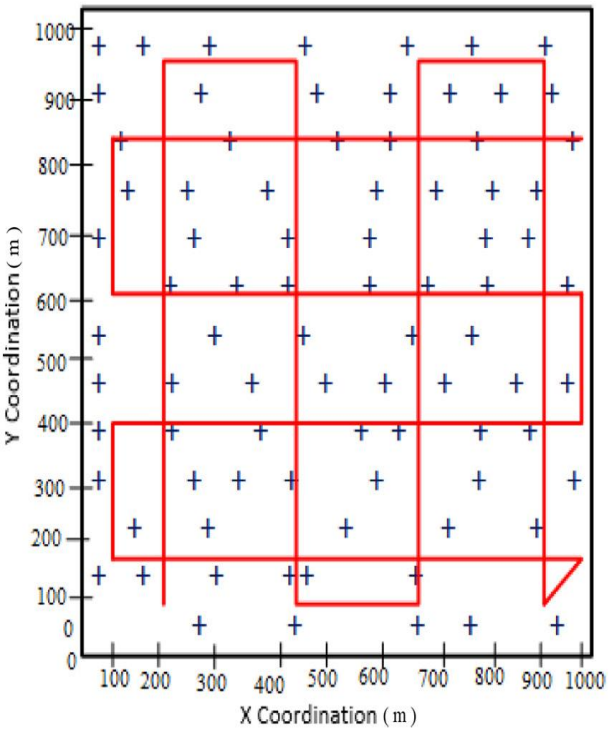

Fig 3.3 Double Scan trajectory

D. Hilbert

Hilbert makes many turns to minimize the collinearity issues without increasing the path length as shown in Figure. However, it also has a potential drawback that is the mobile beacon will never

traverse on the corner of the sensing field. The sensor close to the corner will hear beacon points only from one direction, and their estimate will not be authentic. In Hilbert, the whole network is not covered and have higher chances of path planning error. Hilbert curve divides the two dimensional space into the 4n square cell and connects the center of those cells using 4n line segments. In the equation, 4n is the level of Hilbert curve, and R is the resolution of the trajectory. The entire distance traversed by the mobile anchor node in Hilbert is calculated by

$$D_{Hilbert}=4^n R$$

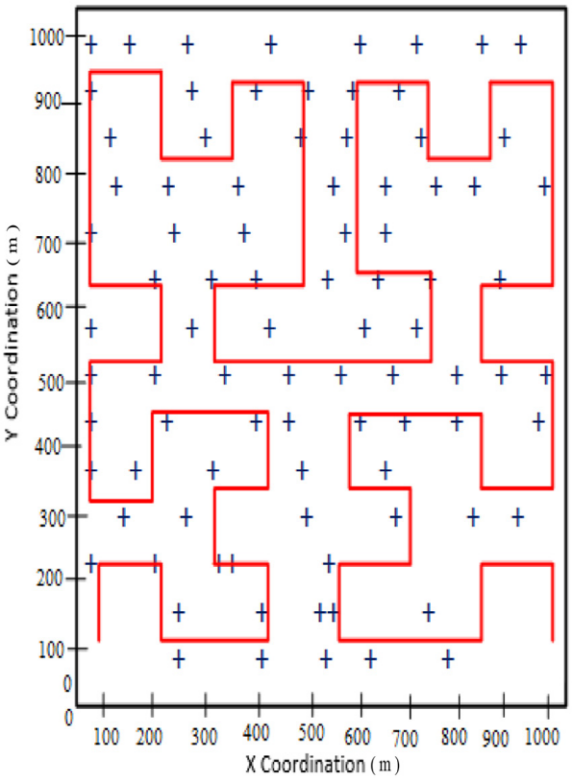

Fig 3.4 Hilbert trajectory

E .Circles trajectory

Circles path planning algorithm is used to reduce the collinearity issues as shown in Figure. Mobile anchor follows circular trajectory instead of a straight line to avoid the collinearity issues. Circle do not cover the whole network effectively. Moreover, Circles has a scalability issue. When the sensing field extends, Circles becomes larger. When circles become larger, collinearity issues raise and hence decreases the path planning accuracy. The resolution d is defined as half of the radius of the innermost circle. Dividing the network size into n3n sub-squares, the total distance traversed by the mobile landmark in Circle can be represented as a function of d and n as Follows

$$D= n^2\pi R/4 + ((n/2) – 1)R$$

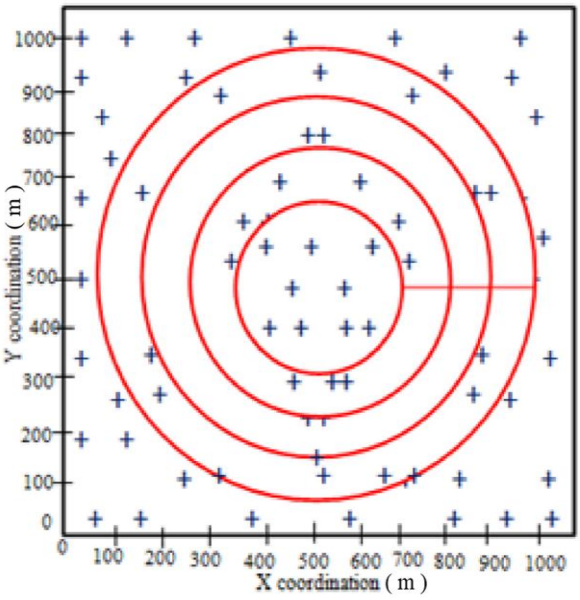

Fig.3.5 circle trajectory

IV. PROPOSED SYSTEM

we consider the structural similarities between the spider-webs and path planning segments and try to create a spider web-like model for path planning to obtain the paths. The spider-web is constructed by the spokes and hypotenuses. Spokes begin from the web center and form the framework of the structure. Hypotenuses are the concentric circles around the web center. The lines formed by road intersections with same layer are regarded as the hypotenuses. The road segments connecting the adjacent layer intersections can be regarded as the spokes

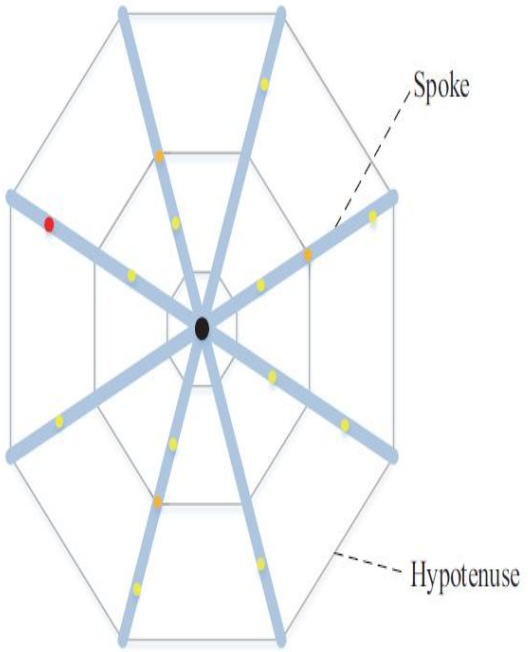

Fig. 3.1 Spider web-like model

We select the source intersection based on the distance between the source vehicle and the candidate intersection, as well as the angle formed by the candidate intersection,. Each candidate intersection has a grade, and the candidate with the highest grade is selected as the first node of path intersection.

The grade expression is as Eq. (1):

$$G(i) = \lambda(1 - \frac{d(i)}{L}) + (1 - \lambda)D(i)$$

Where L is the length of path , d(i) isthe distance between the intersection .direction parameter expressed by Eq. 2

$$D(i) = \begin{cases} 0 & (\alpha > \frac{\pi}{2}) \\ 1 & (\alpha < \frac{\pi}{2}) or (\alpha = \frac{\pi}{2} \& Dir = 1) \end{cases}$$

In Eq. (2), _ is the angle formed by the mobile beacon ,In whichthe NODE is as the vertex, the node intersection are as the end points. Dir is thevehicle movement direction. If the vehicle is moving towardsintersection i, then Dir = 1, otherwise Dir = 0.Without loss of generality, we assume that I1 and I2 are thetwo neighbor intersections of A, _I1AI2 is the angle betweenthe lines which connect A, I1 and A, I2. _D1AD2 is the anglebetween the lines connecting A with D1, D2 in Fig. AI1
and AI2 are defined as the spokes. The condition _I1AI2 and_D1AD2 should satisfy Eq. (3).

$$\theta_{D_1AD_2} \leq \theta_{I_1AI_2\,min}$$

Note that when _D1AD2 = 0, there is only one destinationintersection. Besides, if AI1 and AD1 (AI2 and AD2)coincide, only the intersection I1 (I2) is selected as the firstlayerintersection. In that case there is only one first-layerintersection. Otherwise _I1AI2 min should be the minimumangle bigger than _D1AD2 .

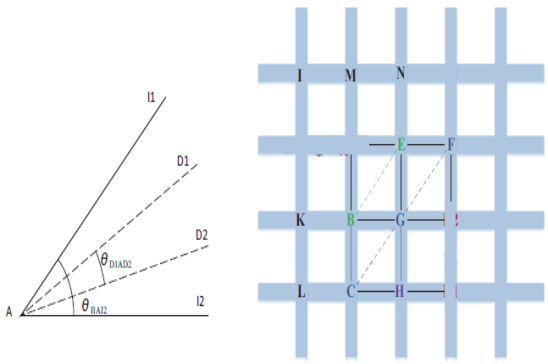

Fig.4.2 (a) First-layer intersection (b) Each layer intersection

We can construct the spider web like model, according to which the available paths consisting of intersections can be found

Input: number of nodes
Output: transmission path
1: procedure path discovery
2: create the spider web like model w
3: create the path tree t
4: depth-first search t and get paths
5: send request spiders
6: activate(request clock)
7: if receive confirmed spider when request clock isnot timeout then
8: calculate intersection of node when receiveconfirmed spider
9: return the path with the smallest (Tr-Ts)
10: else
11: restart Path Discovery
12: end if
13: end if
14: end procedure

A. Network simulation

In communication and computer network research, network simulation is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities(host/routers, data links, packets, etc) using mathematical formulas, or actually capturing and playing back observations from a production network. The behavior of the network and the various applications and services it supports can then be observed in a test lab,

Various attributes of the environment can also be modified in a controlled manner to assess how the network would behave under different conditions. When a simulation program is used in conjunction with live applications and services in order to observe end-to-end performance to the user desktop, this technique is also referred to as network emulation.

B. Network simulator

A network simulator is a piece of software or hardware that predicts the behavior of a network, without an actual network being present.The network simulator is the program in charge of calculating how the network would behave. Such software may be distributed in source form(software) or packaged in the form of a dedicated hardware appliance. Users can then customize the simulator to fulfill their specific analysis needs. Simulators typically come with support for the most popular protocols in use today, such as IPv4,IPv6,UDP, and TCP.

C. Uses of network simulators

Network Simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple

networked computers, routers and data links, network simulators are relatively fats and inexpensive. They allow engineers to test scenarios that might be particularly difficult or expensive to emulate using real hardware – for instance, simulating the effects of a sudden burst in traffic or a DoS attack on a network service. Networking simulators are particularly useful in allowing designers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment.

Network simulators, as the name suggests are used by researchers, developers and QA to design various kinds of networks, simulate and then analyze the effect of various kinds of networks, simulate and then analyze the effect of various parameters on the network performance. A typical network simulator encompasses a wide range of networking technologies and help the users to build complex networks from basic building blocks like variety of nodes and links.Withthe help of simultors one can design hierarchial networks using various types of nodes like computers, hubs, bridges, routers, optical cross-connects, multicast router, mobile units, MSAUs etc.

D. Ns (simulator)

Ns or the Network simulator(also popularly called ns-2) is a discrete event network simulator. It is popular in academia for its extensibility(due to its open source model) and plentiful online documentation. Ns is popularly used in the simulation of routing and multicast protocols, among others, and is heavily used in ad-hoc networking research. Ns supports an array of popular network protocols, offering simulation results for wired and underwater networks alike. It can be also used as limited –functionality network emulator. Ns is licensed for use under version 2 of the GNU General Public License.

E. About tcl

Tool Command Language (TCL) is an interpreted script language developed by Dr.JohnOusterhout at the University of California, Berkely, and now developed and maintained by Scriptics. Tcl is comparable to : Netscape JavaScript Microsoft's Visual Basic .The UNIX-derived Practical Extraction and Reporting Language IBM's Restructured Extended ExecutorIn general,script languages are easier and faster to code in than the more structured,complied languages such as C and C++. Script languages are sometimes considered good "glue"languages for tying several compiled programs together.Or,as stand-alone programs,they can allow you to create simple but powerful effects on their own.TclBlend is a version of Tcl that can access certain Java languages facilities. Tcl has a companion program,Tool kit (Tk), to help create a Graphical User Interface with Tcl.

F. About otcl

OTCL is an object orinted extension of Tcl and created by David Wetherall.It is used in network simulator(NS-2) and usually run under Unix environment.

G. About gedit

GEDIT is a UTF-8 compatible text editor for the GNOME computer desktop environment. Designed as a general purpose text editor,gedit emphasizes simplicity and ease of use.It includes tools for editing source code and structured text such as markup languages.It is designed to have a clean,simple graphical user interface according to the philosophy of the GNOME project ,and it is the default text editor for GNOME.

Gedit includes syntax highlighting for various program code and text markup formats. Gedit also has GUI tabs for editing multiple files.Tabs can be moved between various windows by the user.It can edit remote files using GVFS(Gnome VFS is now deprecated) libraries.It supports a full undo and redo system as well as search and replace.Other typical code oriented features include line numbering,bracket matching,text wrapping, current line highlighting, automatic indentation and automatic file backup.Some advanced features of gedit include Multilanguage spellchecking and a flexible plugin system allowing to dynamically add new features.

V. CONCLUSION

To fulfil the real-time constraints, we proposed new path planning, a novel spider web-like mechanism for localization. We create a spider web-like model to restrict the searching area. Simulation results show that our proposed scheme performs better than conventional in terms of average transmission delay, packet delivery ratio. The results prove that spider mechanism can ensure real-time requirements of in the presence of network congestion

Future work

In future dual mobile beacon will used instead of single in order to reduce path overhead and to achieve higher level of accuracy

REFERENCES

[1] Liu Y, Yang Z, Wang X, et al. Location, localization, and localizability. J Comput Sci Technol 2010; 25(2): 274–297.
[2] Ahmed N, Kanhere S and Jha S. The holes problem in underwater sensor networks: a survey. ACM SIGMOBILE Mob Comput Commun 2005; 9(2): 4–18.
[3] Katiyar V, Chand N and Soni S. A survey on clustering algorithms for heterogeneous underwater sensor networks. Int J Adv Netw Appl 2011; 2(4): 745–754.
[4] Akyildiz IF, Su W, Sankarasubramaniam Y, et al. Underwater sensor networks: a survey. Comput Netw 2002; 38(4): 393–422.
[5] Maraiya K, Kant K and Gupta N. Application based study on underwater sensor network. Int J Comput Appl 2011; 21(8): 9–15.
[6] Arampatzis T, Lygeros J and Manesis S. A survey of applications of underwater sensors and underwater sensor

networks. In: Proceedings of the intelligent control and international symposium on, Mediterranean conference on control and automation, Limassol, Cyprus, 27–29 June 2005, pp.719–724. New York: IEEE.
[7] Huang R and Za´ruba GV. Monte Carlo path planning of underwater sensor networks with a single mobile beacon. Wirel Netw 2009; 15(8): 978–990.
[8] Perrig A, Szewczyk R, Tygar JD, et al. SPINS: security protocols for sensor networks. Wirel Netw 2002; 8(5): 521–534.
[9] Xiang L, Luo J and Vasilakos A. Compressed data aggregation for energy efficient underwater sensor networks In: Proceedings of the 8th annual communications society conference on sensor, mesh and ad hoc communications
[10] and networks, Salt Lake City, UT, 27–30 June 2011, pp.46–54. New York: IEEE.
[11] Lin JW and Chen YT. Improving the coverage of randomized scheduling in underwater sensor networks. IEEE T Wirel Commun 2008; 7(12): 4807–4812.
[12] Zeng Y, Xiang K, Li D, et al. Directional routing and scheduling for green vehicular delay tolerant networks. Wirel Netw 2013; 19(2): 161–173.
[13] Sheng Z, Yang S, Yu Y, et al. A survey on the IETF protocol suite for the Internet of Things: standards, challenges, and opportunities. IEEE Wirel Commun 2013; 20(6): 91–98.
[14] Zeng Y, Cao J, Hong J, et al. Secure path planning and location verification in underwater sensor networks: a survey. J Supercomput 2013; 64(3): 685–701.
[15] Han G, Xu H, Duong TQ, et al. Path planning algorithms of underwater sensor networks: a survey. Telecommun Syst 2013; 52(4): 2419–2436.
[16] Kumar S and Lobiyal DK. Power efficient range-free path planning algorithm for underwater sensor networks. Wirel Netw 2014; 20(4): 681–694.
[17] Camp T, Boleng J and Davies VA. A survey of mobility models for ad hoc network research. Wirel Commun Mob Comput 2002; 2(5): 483–502.
[18] Halder S and Ghosal A. A survey on mobile anchor assisted path planning techniques in underwater sensor networks. Wirel Netw 2016; 22(7): 2317–2336.
[19] Stoleru R and Stankovic JA. Probability grid: a location estimation scheme for underwater sensor networks. In: Proceedings of the 1st annual communications society conference on sensor and ad hoc communications and networks, Santa Clara, CA, 4–7 October 2004, pp.430–438. New York: IEEE.
[20] He T, Huang C, Blum BM, et al. Range-free path planning schemes for large scale sensor networks. In: Proceedings of the 9th annual international conference on mobile computing and networking, San Diego, CA, 14–19 September 2003, pp.1–16. New York: ACM