Tic-Tac-Toe-Arch: A Self-organizing Virtual Architecture for Underwater Sensor Networks

Tamoghna Ojha, Manas Khatua, Sudip Misra

School of Information Technology
Indian Institute of Technology Kharagpur
Kharagpur - 721302, West Bengal, India
{tojha|manask|smisra}@sit.iitkgp.ernet.in

Abstract

In this paper, we propose Tic-Tac-Toe-Arch, an *energy-efficient*, *self-organizing*, and *virtual* network architecture for Underwater Sensor Networks (UWSNs). In UWSNs, underwater currents play a major role for connectivity disruption. The Tic-Tac-Toe-Arch is capable of self-organizing the selection of active nodes to maintain the connectivity providing a *virtual topology*. Further, this selection is such that the redundant nodes are selected based on node density and passive node mobility in the corresponding underwater layer. Thus, the network energy consumption is less due to different, but low activity ratio, through the layers.

Prior works on UWSN architecture, such as Multipath Virtual Sink, 3D Architecture, and EDETA, did not consider the change in topology due to passive node mobility. Moreover, they did not consider the duty-cycle control through topology management, which, in turn consumes high energy. We evaluated the performance of Tic-Tac-Toe-Arch through simulations in NS-3. In terms of the topology formation time, the simulation results show that 99.3% performance improvement is achieved in Tic-Tac-Toe-Arch than EDETA.

I. Introduction

A network architecture specifies the functionalities of network communication framework to design, implement, and perform the specific tasks. UWSNs differ from their terrestrial counterpart in many aspects [1], [2], [3]. The unique characteristics of UWSNs create the necessity of designing a specific network architecture for UWSNs. We discuss two such features, passive node mobility and high resource consumption, which makes network architecture design in UWSNs more challenging.

Passive node mobility due to underwater currents is an inevitable issue in UWSNs. Typically, the mobility of nodes in underwater environments have been reported to be around 2-3 $knots$ or 3-5 km/hr [3], [4]. As a consequence of node mobility, the network topology changes rapidly with the variable time gap from surface to bottom in an underwater column. This spatial variability of the sensor nodes have effect on the connectivity between the nodes and the hop-to-hop data delivery faces temporary losses of connectivity.

Another major issue in UWSNs is high resource consumption, which significantly affects the performance of network protocols, which, in turn, may result in instability in the network. High resource consumption occurs mainly because of high power requirements of the acoustic modem. For example, [5] proposes a modem, which

consumes 0.203 $watts$, 0.024 $watts$ and 3×10^{-6} $watts$ for transmit, receive and sleep, respectively. The overall network energy consumption is also increased, as the communication range of a node is small, and, thus, it requires more number of hops to reach the sink node. More number of active nodes in a network also increases the overall power consumption of the network. Therefore, we minimize the network energy consumption by controlling the activity ratio [6] of nodes and the duration of activity.

There exists three types of architecture in the literature proposed for UWSNs – static, mobile, and hybrid. In static architecture, the deployed sensor nodes do not change their positions with time, i.e., the topology remains fixed. However, this does not reflect the real scenario of UWSNs. In mobile architecture, sensor nodes move freely and topology changes very rapidly. Hybrid Architecture, consists of costly autonomous vehicles and ordinary sensor nodes both. However, all these solutions are inadequate to provide a network architecture which is energy-efficient and adaptive with the changes of topology with time.

In this paper, we present an architecture for UWSNs, named as Tic-Tac-Toe-Arch. Tic-Tac-Toe-Arch forms a virtual architecture from the actually deployed sensor nodes. The architecture is capable of maintaining connectivity from the source nodes to the surface sinks, regardless of the node mobility. This virtual topology is only a part of the original deployment, however, acts as the topology for event reporting of the sensor nodes to the surface sinks. The scheme is capable of self-organizing the selection of active nodes to maintain this virtual topology. At the same time, it reduces energy consumption by maintaining low activity ratio. In this work, we consider a 3D deployment of sensor nodes with multiple sinks floating on the surface. Our proposed architecture does not introduce any additional cost for deployment, as all the nodes, except the sink nodes, are simple in nature, and no super nodes are deployed under water. In brief, our contributions in this work are as follows :

- We propose an algorithm for calculating the duration of connectivity between underwater nodes depending on the speed of the underwater current present in different layers of water.
- We propose a self-organizing network architecture by utilizing the dynamic formation of *virtual topology*. The proposed architecture can address the topology breakup issue due to node mobility.

The rest of this paper is organised as follows. In Section II, we discuss the background and related deployment architectures. Tic-Tac-Toe-Arch, the proposed architecture, is explained in Section III, and the simulation results are discussed in Section IV. Finally, we conclude the paper in Section V.

II. Related Works

Pompili and Melodia proposed a static architecture [7] for ocean bottom monitoring applications. In this architecture, sensor nodes were bottom anchored and multiple sinks were deployed in the water surface. However, this type of deployment requires long range communication between sea-bed sensor nodes and surface sinks. Moreover, this type of architecture can only cover the ocean bottom. Improving this architecture, in [8], and [9], authors further analyzed how to achieve maximum sensing and communication coverage with minimum number of sensor nodes deployed. Three strategies were proposed to ensure the optimal sensing and communication coverage: 3D-random, bottom-random, and bottom-grid. Following the same objective to cover the 3D ocean column, different

architectures were proposed in [1] and [10]. In the architecture proposed by Cayirci *et al.* [10], sensor nodes were lowered to different depth via a cable from the surface buoys and the cable was used to communicate between the surface buoy and the sensors. Another architecture for wide area ocean networks was introduced by Roy *et al.* in [11], where a set of sensor nodes were tethered by cables, and supported by a backbone network. However, this additional cables adds extra implementation cost to network, and long cables are required to cover deep-water regions which increases the complexity of the network.

One hierarchical network architecture, Multipath Virtual Sink architecture [12], was introduced to reduce the loads of sink nodes and to increase robustness in a network. In this work, sensor nodes were grouped into few clusters and each cluster has one local aggregation point called the *virtual sink*. The authors assumed that these virtual sinks together forms a mesh network that connects to the local sinks and can communicate using high speed links. Use of virtual sinks reduces the load of sink nodes and the use of multipath data delivery scheme increases data redundancy.

EDETA [13], a routing protocol adapted to UWSN, proposed a UWSN architecture in which sensor nodes arrange themselves in several clusters, and one of them decides to be the cluster head (CH), based on a calculated threshold value. CHs of each cluster form a tree structure to forward the data to the sink node. However, the work is silent about how the cluster structure is maintained in the presence of node mobility. Additionally, the time taken to completely form a network structure is high due to lots of message transfer between the nodes and the corresponding CHs. The higher the number of message exchange, the higher is the overall energy consumption of the network.

All these architectures are static in nature, e.g., the topology does not change with time. Unfortunately, this assumption will not work considering the actual scenario in a UWSN.

The importance and research challenges of mobile architecture for UWSNs was discussed in [3]. The authors argued about the architectural requirements for long-term non time-critical, and short-term time-critical applications in UWSNs. Further assumption was, that the sensor nodes are capable of relocating themselves, if required. Wang *et al.* proposed such type of network using mobile actors and sensor nodes [14]. Actors dive to different depths and the sensor nodes move accordingly to form temporary clusters, called virtual clusters, in which the actors act as cluster heads to collect the data from all the sensor nodes, and finally send data to the base station. Although, in these works, the sensor nodes are free to move, the mobility profile assumed does not match with practical UWSNs. Moreover, the inclusion of Autonomous Underwater Vehicles (AUVs) or Remotely Operated Vehicles (ROVs) make the network implementation expensive.

In [15], Zhou *et al.* proposed a localization scheme considering a UWSN architecture with three types of nodes: surface buoys, anchor nodes and ordinary nodes. The authors assumed that the anchor nodes are special nodes that can directly communicate with the surface buoys. However, this type of special nodes will require a modem for long distance acoustic communication which will consume more battery power and will increase the deployment cost.

We compare the different architectures and categorize them according to various features in Table I.

TABLE I: Comparison of the existing architectures according to different features

	Type	**How Deployed?**	**Passive Node mobility**	**3D/2D**
Cayirci *et al.* [10]	Static	Cabled from buoy	Not considered	3D
Pompili *et al.* [9]	Static	Floating	Not considered	3D
Roy *et al.* [11]	Hybrid	Cable back-boned	Not considered	3D
Multipath Virtual Sink architecture [12]	Static	Floating	Not considered	2D
Wang *et al.* [14]	Hybrid	Self movement	Not considered	3D
EDETA [13]	Static	Floating	Not considered	2D

III. Tic-Tac-Toe-Arch

A. Brief Overview

The design goal of the Tic-Tac-Toe-Arch is to provide an efficient architecture that can provide communication guarantee from source nodes to surface sinks, provided that the sensor nodes (source and forwarder) are not isolated. At the same time, this architecture schedules the nodes in such a way that energy consumption of the network is minimized by maintaining a low activity ratio. We propose an intelligent sleep-wakeup scheduling algorithm that helps to improve the network lifetime by allowing redundant nodes to stay in the sleeping state. The proposed algorithm determines the redundant nodes according to the node mobility in different levels. Therefore, at any time, a larger proportion of the nodes are in the sleep state and the rest are in the active state. If we denote an active node by 'O' and a sleeping node by 'X', then the side view of the architecture looks like a grid pattern of 'O' and 'X', in which one row looks like O-X-O-X, visually similar to that of Tic-Tac-Toe game. The number of 'X' between two successive 'O' may be different according to the node density and the speed of node movement in that level. Following this, we named the architecture as "Tic-Tac-Toe-Arch".

The salient features of the Tic-Tac-Toe-Arch, which differentiates it from the existing architectures of UWSNs are as follows:

a. *It is a virtual architecture for both 2D and 3D deployment.* This architecture is usable in networks formed using any random node deployment methodology. The only prerequisite of this scheme is that the nodes are aware of their own location and have a knowledge about the location of the sink node.

b. *It is self-configurable under mobile environment.* Tic-Tac-Toe-Arch enables the underwater sensor nodes to take intelligent decision on neighbour selection, which helps to cope with the dynamic changes in network topology.

c. *Source to sink connectivity is always maintained, provided that the nodes are not isolated with respect to their transmission range.* The proposed intelligent neighbour selection method helps to maintain inter-node connectivity by choosing an appropriate neighbour. From the underwater source nodes to the surface sinks, this selection method is performed iteratively and, thus, the connectivity between the underwater nodes is preserved.

d. *Architecture formation time is very less.* Our experimental results show that the time required for forming a *virtual topology* to maintain node-to-node connectivity between the sensor nodes is $O(n)$, where, n is the number of neighbours of a node. For a network with total N number of nodes, $n \leq N - 1$.

e. *It is scalable to the number of nodes.* Even if we change the number of deployed nodes in the network at any time, the proposed architecture is scalable to adapt the change in the subsequent cycle.

B. Assumptions

Our deployment scenario considers two types of nodes in the network, namely – the ordinary and the sink nodes. The ordinary nodes are simple nodes with limited battery power and less computation capability. On the other hand, the sink nodes are empowered with more battery power and complex computation capability. The ordinary nodes generally remain submerged in water, whereas the sink nodes generally float on the surface of the water. For example, we assume an ocean environment monitoring application, which requires sensor nodes to be deployed throughout the target area and the sink nodes to be on the surface. Some sensor nodes are anchored with the sea-bed as well. The sink nodes are capable of using radio frequency to communicate with other sink nodes or base station, and uses acoustic signal to communicate with the ordinary nodes which remain submerged in water throughout the ocean. A network model of the above mentioned scenario is depicted in Figure 1. However, the sensor nodes need not to be time-synchronized with one another.

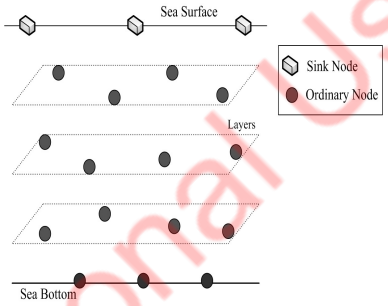


Fig. 1: 3D underwater deployment scenario

C. Virtual Topology Formation

The detailed functionality of the Tic-Tac-Toe-Arch method is discussed below. The architecture formation process consists of three phases – *Neighbour finding*, *Best neighbour selection* and *Set duty cycle*. Once the architecture is configured, nodes are ready for data transmission.

1) Neighbour Finding: To provide the connection from the source nodes to the surface sinks, we should ensure that each sensor node has at least one neighbour in its communication zone, which will help to forward the sensed information towards the surface sink. The basic operations of neighbour finding phase is shown in Algorithm 1. At first, the seabed sensors broadcast 'REQ' message and discover their neighbours after receiving the 'RPLY' messages. To restrict unnecessary message flooding, only the sensor nodes, which are closer to the sink than the present sender node, replies back to the sender. For example, in 3D deployment, depth information may be considered as a qualifying parameter for sending the 'RPLY' message against a broadcast message. Let, the two sensor nodes which sent 'REQ' and received 'REQ' are i (depth = d_i) and j (depth = d_j), respectively. Here, j will send 'RPLY' to i only if it is at less depth i.e. $d_j < d_i$. These nodes which send 'RPLY' form the set of effective neighbours ($\mathcal{N}_{eff}(i)$) of the sender (i).

A node waits $RPLY_{timeout}$ time for receiving all the 'RPLY' messages. In Equation 1, we show how a node calculates $RPLY_{timeout}$. It is the sum of round trip time (twice the propagation delay) with additional time required for sending ($REQ\_sending\_delay$) and processing ($REQ\_processing\_delay$) the 'REQ' message, and sending 'RPLY' ($RPLY\_sending\_delay$) message. We have,

$$RPLY_{timeout} = 2 \times t_{prop} + REQ\_sending\_delay + REQ\_processing\_delay + RPLY\_sending\_delay \quad (1)$$

where, t_{prop} is the propagation delay between nodes i and j. It is the ratio of the transmission range (R) of a sensor node to the average sound velocity (V_s), and is calculated by Equation 2.

$$t_{prop} = \frac{R}{V_s} \quad (2)$$

After the $RPLY_{timeout}$ time elapses, the next phase of the Tic-Tac-Toe-Arch, *best neighbour selection*, is executed.

Algorithm 1: Neighbour Finding for node i and neighbour j

```
if Node i type = 'Source' or 'Selected' and state = 'Undecided' then
    if N_eff(i) = NULL then
        Node i broadcasts a 'REQ' message;
        time = T_1;
    if Node i received a 'RPLY' message then
        time = T_4;
    if RPLY_timeout elapsed then
        Node i executes 'Best neighbour selection' algorithm;
    else
        Node i waits for 'RPLY' message;
else
    if Node j state = 'Dormant' and received a 'REQ' message then
        time = T_2;
        if d_j < d_i then
            Send 'RPLY' message back to node i;
            time = T_3;
```

2) Best Neighbour Selection: We need to choose a neighbour that has the ability to provide connectivity for maximum duration to the sender node (i). For each neighbour node $j \in \mathcal{N}_{eff}(i)$, we calculate the *time-to-disconnect* (t_d) value, which is the duration of time a neighbour will take to move out of the range of a node. The calculation of t_d is discussed in Section III-D3. In Algorithm 2, we show the steps of *best neighbour selection*. The maximum t_d value is calculated among all the t_d values using the function *Calculate-ToD()*. The neighbour with maximum t_d value is the best candidate among all the neighbours, and is thus tagged as the 'Selected node' or the 'Virtual node'. After this phase, the next phase *set duty cycle* is executed.

3) Set Duty Cycle: After selecting a neighbour as the best candidate, a node gets a connectivity towards the sink, for t_d time. Other neighbours, except the selected one, will be redundant during this period. These nodes are transitioned to the sleep mode for t_d time, that is, until when the selected neighbour moves out of its communication zone. A command message 'CMND' is broadcast from the node to let its neighbours know about the scheduling.

Algorithm 2: Best Neighbour Selection
GetMaxToD() returns the maximum t_d value

```
if Node i type = 'Source' or 'Selected' and state = 'Undecided' then
    foreach 'RPLY' message receive do
        t_d = Calculate-ToD();
    T_MAX = GetMaxToD();
    foreach j ∈ N_eff(i) do
        if t_d == T_MAX then
            Set node j type = 'Source' & state = 'Active';
            Set node j type = 'Selected' & state = 'Undecided';
```

This sleep-wakeup schedule of nodes, helps in maintaining low activity ratio in the network, thereby helping in minimizing energy consumption in the network. The more the energy saved by the nodes, the more the network lifetime. After the t_d time elapses the method is repeated and again a new neighbour is selected. The t_d time varies along the ocean column due to the existence of different mobility profiles in the ocean column. The higher the node mobility, the lesser the value of t_d. Algorithm 3 outlines the operations of this phase.

Algorithm 3: Set Duty Cycle

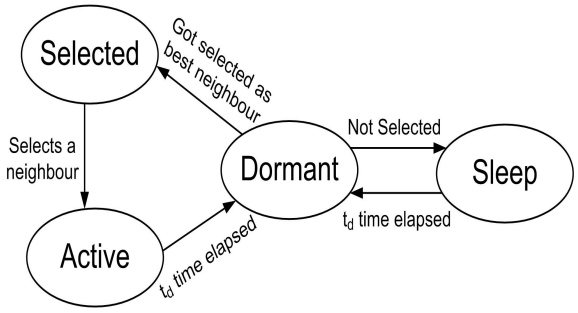
```
if Node type = 'Source' and Node state = 'Active' then
    Send 'CMND' message;
if received a 'CMND' message then
    if Node type = 'Selected' and Node state = 'Undecided' then
        Ready to receive data;
    else
        Node state = 'Sleep';
if t_d time elapsed then
    Set all node's & neighbour's state = 'Dormant';
    Execute 'Neighbour Finding' algorithm;
```

D. Discussions

1) Types and states of node: In the proposed architecture, a sensor node undergoes four different states: 'Dormant', 'Selected', 'Active', and 'Sleep'. The 'Dormant' state indicates that the node has not undergone the architecture configuration phase at all or is just waiting for undergoing it once again. A node remains in the 'Selected' state until the completion of the architecture configuration and finally it changes its state to either 'Sleep' or 'Active', depending upon the outcome of the configuration phase.

The state diagram of an ordinary sensor node is described in Figure 2(a). All nodes except the source nodes remain at 'Dormant' state initially. Once a node gets selected as the best neighbour, it changes its state to 'Selected'. All the other nodes which are not selected change to the 'Sleep' state for previously mentioned 'time-to-disconnect' period. The selected node again repeats the procedure of neighbour finding as its previous node. After selecting a neighbour, it changes to the 'Active' state. These nodes change their state again to 'Dormant', when the 'time-to-disconnect' time elapses.

2) The Control Message Formats: Three types of control messages are exchanged between nodes during the execution of the Tic-Tac-Toe-Arch. These control messages are: Request (REQ), Reply (RPLY), and Command



(a) State diagram of an ordinary node

Type	Node depth

REQ packet

Type	REQ receiving time	RPLY sending time	Average velocity

RPLY packet

Type	Best neighbour id	2nd Best neighbour id	Time-to-disconnect

CMND packet

(b) Different control message formats

Fig. 2: State diagram and Different packet formats

(CMND). Figure 2(b) illustrates the format of these messages. A sender node broadcasts 'REQ' to discover its neighbours, who send 'RPLY' back to it. A 'CMND' message is sent from a node to inform its neighbours about their future scheduling.

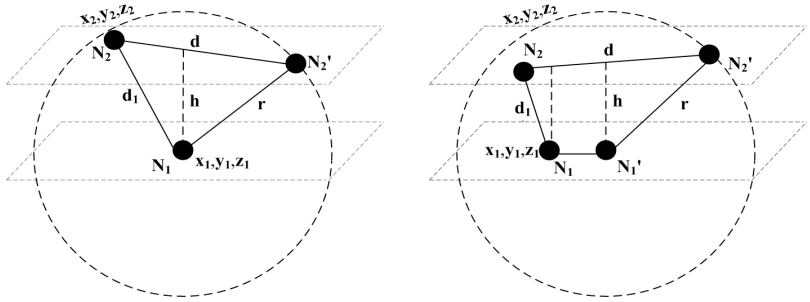
3) Calculation of Duration of Connectivity: The duration of connectivity or 'time-to-disconnect' (t_d) is measured using the *Calculate-Td()* function used in algorithm *best neighbour selection*. As some of the sensor nodes are stationary and some are mobile, we consider two different scenarios. Figure 3(a) depicts the scenario in which one node is static and another is mobile, and Figure 3(b) depicts the scenario where both the nodes are mobile. Before calculating t_d, it checks for the values of its own and its neighbours mobility, which is extracted from the 'RPLY' packet.

Let us consider that a node N_1 broadcasts the 'REQ' message at $t = T_1$, and a neighbour node N_2 receives the message at $t = T_2$. At $t = T_3$, N_2 replies back with the 'RPLY' message and the reply is received by N_1 at $t = T_4$. T_1 and T_4 are measured according to the local clock of N_1 and T_2 and T_3 are measured according to the local clock of N_2. If δ is the clock drift between the two local clocks of these two nodes, then $T_2 = T_1 + \delta + t_{prop}$ and $T_4 = T_3 - \delta + t_{prop}$. Thus the propagation delay between these nodes is $t_{prop} = \frac{(T_2 - T_1) + (T_4 - T_3)}{2}$. Therefore, any clock drift between the sender and the receiver is nullified in the measurement of propagation delay.

In the first scenario, in Figure 3(a), N_1 is a seabed-embedded stationary node, and N_2 a floating node moving with velocity v_m due to the effect of underwater current. The dashed circle represents the transmission zone of N_1. d is the displacement of node N_2 due to underwater current at t_d time, before it moves out of the communication range of node N_1. Therefore, $d = v_m \times t_d$. Again, $d = \sqrt{{d_1}^2 - h^2} + \sqrt{r^2 - h^2}$, as per Figure 3(a), where, $h = |z_2 - z_1|$ is the difference in depths between these nodes and r is the transmission range of N_1. We calculate $d_1 = V_s \times t_{prop}$, where V_s is the average sound velocity in seawater. So, the 'time-to-disconnect' for N_2 is calculated as $t_d = \frac{\sqrt{{d_1}^2 - h^2} + \sqrt{r^2 - h^2}}{v_m}$.

Another scenario, which is shown in Figure 3(b), arises when both the nodes are mobile. The dashed circle represents the transmission zone of N_1 when the node is located at $N_1{}'$, the shifted position due to the effect of passive node mobility. Here, the displacement of node N_2 due to underwater current is $d = v_{m2} \times t_d$. Again, from Figure 3(b), we have $d = \sqrt{{d_1}^2 - h^2} + v_{m1} \times t_d + \sqrt{r^2 - h^2}$, where v_{m1} and v_{m2} are the movement speeds of node N_1 and N_2, respectively. So, the 'time-to-disconnect' for node N_2 in this scenario is calculated as

$t_d = \frac{\sqrt{{d_1}^2 - h^2} + \sqrt{r^2 - h^2}}{|v_{m2} - v_{m1}|}$.



(a) One node is mobile and another is static (b) Both the nodes are mobile

Fig. 3: Different scenarios between nodes

4) Complexity Analysis: We calculate the time complexity of Tic-Tac-Toe-Arch with respect to the number of nodes present in the network.

During the *neighbour finding* phase, a node broadcasts a 'REQ' message and waits for all the 'RPLY' messages for $RPLY_{timeout}$ period. We know, from Equation 1 and 2 that $RPLY_{timeout}$ is constant as $PropagationDelay_{max}$ throughout the network is a constant term. Therefore, the time required to complete this phase of Tic-Tac-Toe-Arch is $O(1)$.

In the second phase, *best neighbour selection*, a node calculate the 'time-to-disconnect' (t_d) value for each 'RPLY' message received. If the number of 'RPLY' message received is n and k_1 is the constant time required to calculate t_d from a 'RPLY' message. Then, total calculation of all the t_d values require $n \times k_1$ unit of time $\equiv O(n)$. Again, the maximum t_d value is found comparing all the t_d values. It also takes $O(n)$ time. Finally, finding the node which has the maximum t_d value also takes $O(n)$ time. Therefore, the overall time complexity of this phase is $O(n)$.

The last phase, *set duty cycle*, again results in constant time for execution. A node commands all its neighbours by sending a 'CMND' packet. It takes constant time. Neighbour nodes receive this message and take necessary action. For this, maximum $PropagationDelay_{MAX} + CMND\_processing\_delay$ time is needed. Both the quantities are constant in this case. Thus, the execution time of this phase is $O(1)$.

Hence, the total time complexity of Tic-Tac-Toe-Arch is the sum of the complexities of these three phases, which is equal to $O(1) + O(n) + O(1) \equiv O(n)$. Here, n is the number of neighbours of a node. This parameter depends on the node density in an area. The more the node density, the more the number of neighbours of a node. Therefore, we conclude that in a dense network, the time required to execute this method will be more. The estimation of node density considering equal node distribution in the network is presented below.

In any deployment scenario, given the number of nodes and area boundary, we can calculate the value of n. Let, N nodes be deployed in a cubic area of side l meters. Considering equal distribution of nodes in the deployment area, the number of nodes present in an unit volume is $\frac{N}{l^3}$. Let, the maximum communication range of each node be r. Then, the volume covered by each node is $\frac{4}{3}\pi r^3$. So, the number of nodes present in this volume is $(\frac{4}{3}\pi r^3) \times (\frac{N}{l^3})$ and each node can have average $n = (\frac{4}{3}\pi r^3 \times \frac{N}{l^3}) - 1$ number of neighbours present in their range. At any time

TABLE II: Simulation Parameters

Parameter	**Value**
Transmission Range (r)	100 m
Node mobility (v_m)	2 - 10 m/s
Node mobility model	Meandering Current Mobility model [17]
Channel frequency	22 KHz
Modulation technique	FSK
Data rate	500 bps
Packet interval	100 s
Speed of sound	1500 m/s
Wave propagation model	Thorp's propagation model [18]
Transmission power	0.203 $watts$ [5]
Receive & Idle power	0.024 $watts$ [5]
Sleep power	3×10^{-6} $watts$ [5]
Initial energy of a node	150 J
Initial energy of a sink	1000 J

in the network, $n \leq N - 1$, that is, $\frac{4}{3}\pi r^3 \times \frac{N}{l^3} \leq N$.

IV. Results and Discussions

A. Simulation Scenario and Parameters

We simulated Tic-Tac-Toe-Arch in NS-3 simulator [16] using the simulator's UAN model. We took four different areas – $100 \times 100 \times 100$ m^3, $150 \times 150 \times 150$ m^3, $200 \times 200 \times 200$ m^3 and $250 \times 250 \times 250$ m^3 in our simulation. In each simulation area, we deployed 50, 100, 150 and 200 nodes, with 5 sink nodes at the surface. This gave us 16 different scenarios to test. All the nodes, excluding the sink nodes, were deployed in multiple layers from sub-surface to sea bottom and randomly placed in each layer. Sink nodes float on water surface and they were also randomly placed. The vertical column was divided in several layers and nodes were placed randomly in each horizontal layer. Each simulation was executed several times and the mean value was plotted. Other simulation parameters are described in Table II.

B. Performance Metrics

We evaluated the performance of our algorithm using the following metrics:

- *Architecture configuration time:* It is the time required to form the *virtual topology* from the deployment of sensor nodes. After the architecture is formed, the nodes can successfully send their information to the sink nodes.
- *Architecture configuration energy:* The energy consumption during the configuration and maintenance of the *virtual topology*.
- *Energy consumption for Data Transmission:* We measured the energy consumption of the sensor nodes due to data packet transmission from source nodes to surface sinks. Reduced energy of a protocol will result in greater lifetime of the network. As node's energy is a critical factor in UWSNs, protocols with lesser energy consumption are required for these networks.

- *Network lifetime:* Network lifetime is the measure of the sustainability of a network over long duration of time. The more the energy spent by the nodes, the less the lifetime of a network. It is a critical measure for different applications in energy constrained networks such as UWSNs. For example, an oceanographic data collection application or an ocean pollution monitoring application needs to sustain for few days or weeks.
- *Time-to-disconnect:* Each node, starting form the source nodes, executes the procedure once, to get a neighbour which will provide it better connectivity. Due to the passive node mobility, these nodes can not remain connected for all the time. Therefore, it is required to rerun the procedure to find the next node which provides better connectivity. The time taken before repeating the procedure is measured by the duration of 'time-to-disconnect' period of the selected neighbour.
- *End-to-end delay:* After the architecture is configured, the nodes start data exchange between them. The metric calculates the difference between the time when source nodes start to send data and when data is received at the sink node.

C. Benchmark

We compare the performance of our protocol with Energy-efficient aDaptive hiErarchical and robusT Architecture (EDETA) [13], a hierarchical routing protocol adapted to UWSN. The execution of the protocol is divided into two phases – initialization phase and normal operation phase. During the initialization phase, sensor nodes arrange themselves in clusters and some of them act as Cluster Head (CH). A tree structure is formed with all the CHs to collect and aggregate data from the other nodes to the sink node. In the normal operation phase, nodes send their data periodically to their CHs, and the CHs send their data to their parents until the data reaches the sink.

EDETA is the most recent work we found in the same aspect with our work. It also helps to minimize the nodes energy consumption by maintaining a synchronized schedule between themselves. The authors justify EDETA protocol by means of extensive simulations in NS-3. Therefore, we choose EDETA as the benchmark to compare with our protocol.

D. Results and Analysis

1) Architecture Configuration Time: Figure 4 shows the average value of the architecture configuration time with different number of nodes in different areas. We can conclude from the results that with increase in the number of deployed nodes, the architecture configuration time also increases. However, with an increase in the deployment area with fixed number of nodes, the architecture is configured in less time. We explain this behaviour with the help of node density. Node density increases when node count in a deployment area increases, and the node density decreases when the same number of nodes are deployed over a large area. An increased number of nodes in a small area means a node is likely to have more number of neighbours in its communication range. More number of neighbours will require more time for execution of the Tic-Tac-Toe-Arch in each node. Therefore, if we increase the deployment area with the same number of nodes, the architecture takes less time to configure, and if we deploy more number of nodes in an area, the architecture configuration time increases.

However, in case of EDETA, the initialization phase, before the nodes can perform normal data transmission, requires fixed period of time 18,000 $seconds$, which is quite high. Comparing with EDETA, we achieved 99.3% performance improvement.

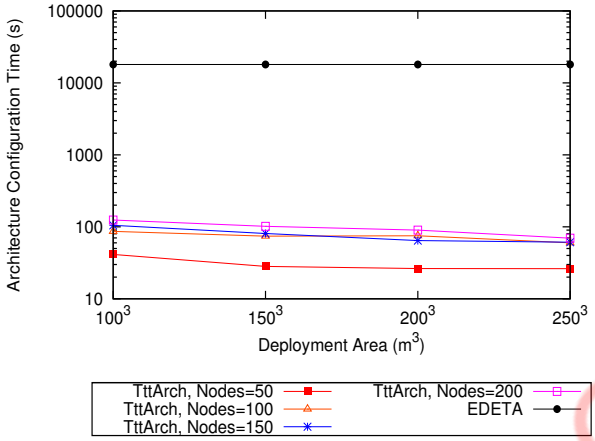
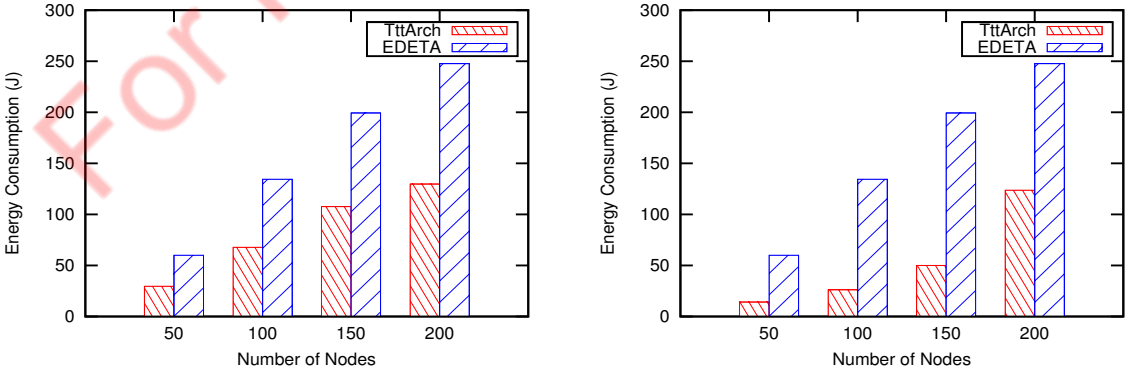


Fig. 4: Architecture Configuration Time

2) Architecture Configuration Energy: The energy consumption during the architecture configuration of Tic-Tac-Toe-Arch and EDETA are shown in Figure 5(a). In this experiment, we varied the number of nodes from 50 to 200 in the simulation area of 100 m × 100 m × 100 m, and plotted the energy consumption to configure the architecture. Results show that the architecture formation in Tic-Tac-Toe-Arch is more energy efficient than that of EDETA. In EDETA, the cluster formation process and determination of data sending schedule of nodes require more number of control message exchange between the nodes and the cluster heads.

We also calculate the energy consumption to maintain the architecture configuration. In Tic-Tac-Toe-Arch, the architecture is dynamically maintained, while in EDETA the cluster formation is done again. The results of this experiment is shown in Figure 5(b). We can conclude from these results that dynamically managing the topology, used in Tic-Tac-Toe-Arch, is more energy efficient than to recreate the architecture again as followed in EDETA.



(a) Energy consumption to Create the Architecture

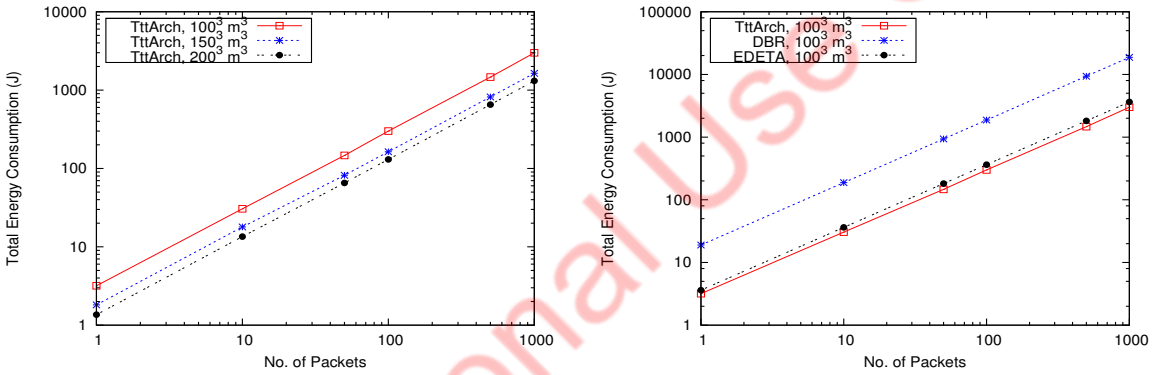
(b) Energy consumption to Maintain the Architecture

Fig. 5: Energy Consumption for Architecture Configuration

3) Energy Consumption for Data Transmission: In Figure 6, we present the results of energy consumption of the network when the number of data packets to be forwarded is varied. In Figure 6(a), we present the results

for energy consumption of Tic-Tac-Toe-Arch at different sized deployment area. We compare the performance of our protocol with Depth Based Routing (DBR) [19] protocol and the EDETA protocol. These results are shown in Figure 6(b). DBR is a very popular and simple routing protocol available for UWSNs. The results show that Tic-Tac-Toe-Arch achieves 83.2% energy efficiency over DBR. However, both Tic-Tac-Toe-Arch and EDETA show nearly similar behaver in this experiment.

In Tic-Tac-Toe-Arch, a node routes the data packets through the 'best neighbour' only. All other neighbour nodes remain in the 'sleep' mode in the meantime. Therefore, sending a packet from sea-bed to surface requires very few hops, which consumes very less energy. On the other hand, in DBR, packets are forwarded to more than one node at a time based on depth information, and this increases the energy consumption significantly. On the other hand, in EDETA, data are sent from a sensor node to the sink following the cluster heads. This also results in limited number of hops towards the sink. Thus, the energy consumption for both Tic-Tac-Toe-Arch and EDETA show similar patterns, and these are less than DBR.



(a) Energy Consumption for Tic-Tac-Toe-Arch (b) Energy Consumption for Tic-Tac-Toe-Arch, DBR, EDETA

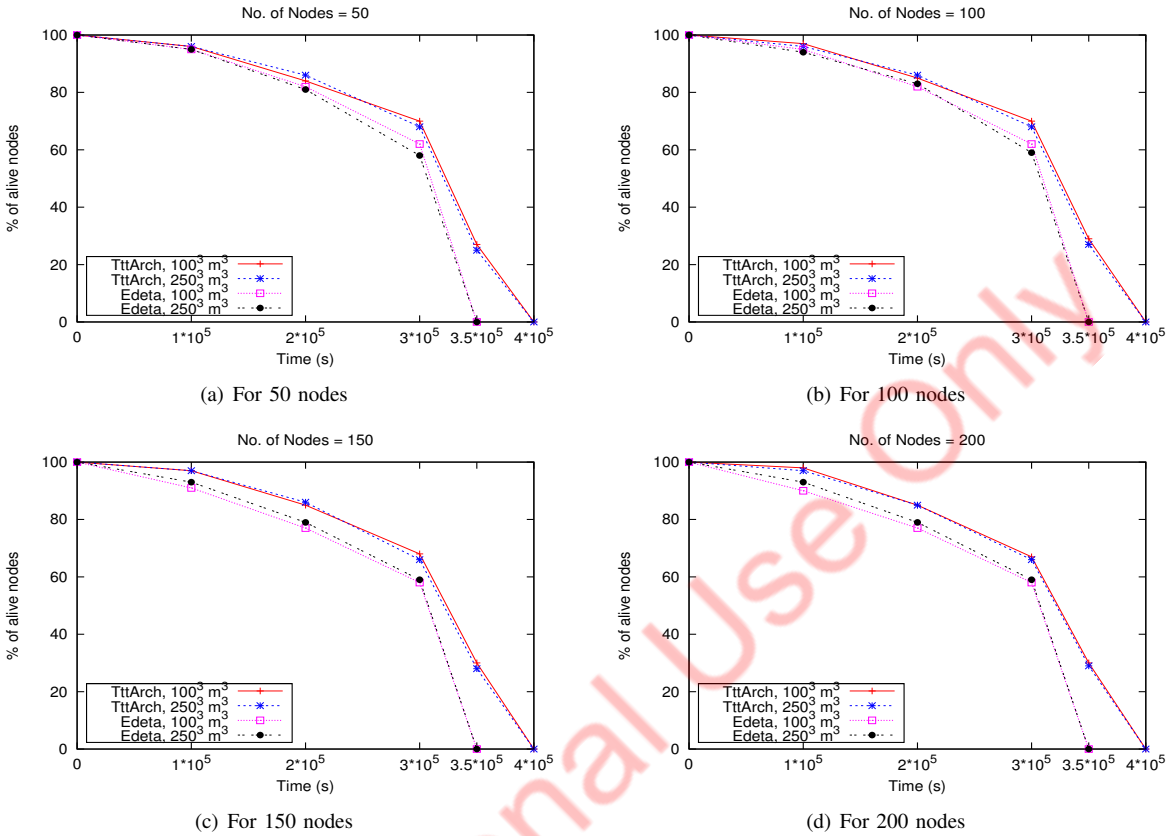
Fig. 6: Energy Consumption for Data Transmission

4) Network Lifetime: We measured the network lifetime for various deployment areas with different node counts. Results are shown in Figure 7. The network lifetime achieved shows the applicability of the architecture for applications such as ocean environment monitoring, which requires a network to be functional for a few days.

During the formation of *virtual topology*, all nodes wake-up, and select the active nodes for the remaining time. By doing this, Tic-Tac-Toe-Arch allows very large percentage of nodes to remain in the 'sleep' state during the data transmission phase. On the other hand, EDETA has a very large architecture configuration time of 18,000 seconds. The cluster structure is formed during this time, and cluster heads (CHs) are selected. Once the clusters are formed, the nodes send their data to the respective CHs, and CHs send their data to their parents. The architecture configuration phase is repeated in EDETA, while the virtual topology is managed dynamically in Tic-Tac-Toe-Arch. Compared to Tic-Tac-Toe-Arch, the architecture configuration, maintenance, and data transmission require more battery power in EDETA. Thus, the proposed architecture saves more energy than EDETA, which results in increased network lifetime in long run.

However, the network lifetime also increases slightly with the increase of the number of nodes (50, 100, 150, 200) in the simulation area (100 m^3). This fact can be explained considering the effect of node density. The

'best neighbour' selection scheme allows other nodes to be in the 'Sleep' state for the rest of 'time-to-disconnect' period. Thus, the overall network energy consumption minimizes with the increase of node density, as less number of nodes remain active.



(a) For 50 nodes

(b) For 100 nodes

(c) For 150 nodes

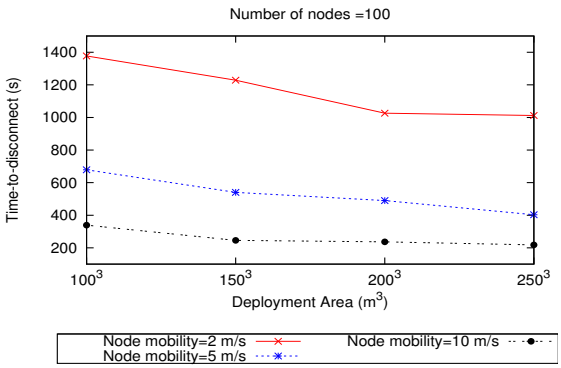
(d) For 200 nodes

Fig. 7: Network Lifetime

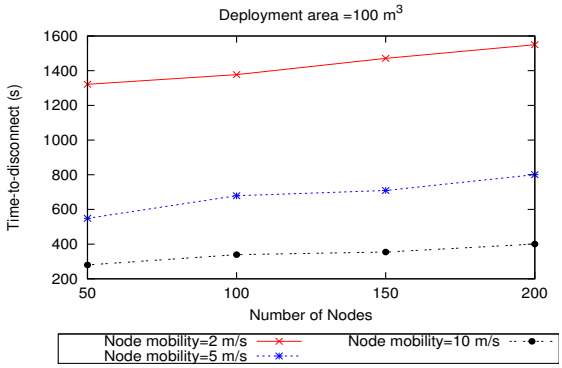
5) Time-to-disconnect: The measure of time-to-disconnect (t_d) indicates the average time duration before the sensor nodes repeat the procedure of *virtual topology* formation. We measured the t_d values for different node count and different deployment area, varying the passive node mobility from 2-10 m/s.

In Figure 8(a), we present the variation of average time-to-disconnect according to different node mobility value. In this scenario, 100 sensor nodes were deployed in different deployment areas – $100 \times 100 \times 100$ m^3, $150 \times 150 \times 150$ m^3, $200 \times 200 \times 200$ m^3, $250 \times 250 \times 250$ m^3. Results show that, t_d decreases with increase of passive node mobility. To study the effect of node density in the measured t_d value, we plotted the t_d increasing the number of nodes in a deployment area of $100 \times 100 \times 100$ m^3. The results of this experiment is shown in Figure 8(b). We conclude, that, in highly mobile underwater networks, the execution of the *virtual topology* formation algorithm is more frequent. Moreover, as the node density increases, the t_d value also increases, which signifies the stability of the formed *virtual topology*.

6) End-to-End Delay: The end-to-end delay in data transmission from any sea-bed node to surface sink is shown in Figure 9. Results show that end-to-end delay increases with increase in the size of the simulation area and total number of nodes. For applications such as ocean monitoring, and habitat monitoring, this order of delay does not affect the performance of the application. However, this amount of delay is not suitable for time critical



(a) Average Time-to-disconnect vs Deployment Area, Number of nodes=100



(b) Average Time-to-disconnect vs Number of Nodes, Deployment area=100m×100m×100m

Fig. 8: Average Time-to-disconnect Value at Different Node Mobility

applications where the time required to report to the sink should be as low as 1 *second*.

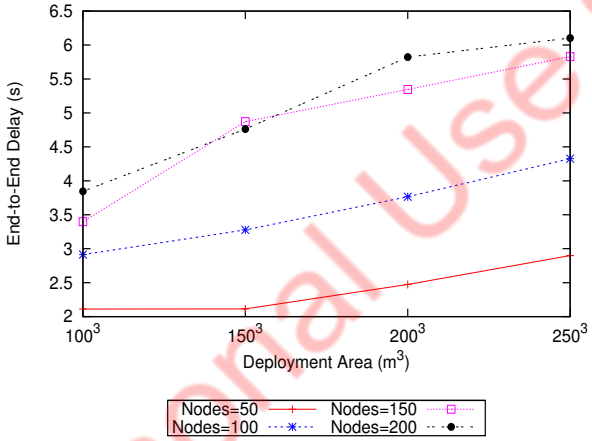


Fig. 9: End-to-End Delay

V. Conclusion

In this paper, Tic-Tac-Toe-Arch, a 3-dimensional network architecture for UWSNs is presented. Tic-Tac-Toe-Arch has *self-organizing* nodes affected by passive node mobility and forms *virtual topology* to maintain node-to-node connectivity. The *virtual topology* is formed by selecting some nodes as active, and the others as redundant, based on the node density and passive node mobility of the corresponding underwater layer. In addition, we used an intelligent sleep-wakeup scheduling methodology to maintain low activity ratio among nodes.

The simulation results show that our method forms the architecture faster compared to the EDETA protocol. The proposed architecture also provides a prolonged lifetime compared to EDETA. We observed the data routing component of the proposed architecture and compared the results with the existing DBR protocol [19], and was seen to be energy-efficient compared to DBR. However, only the active nodes remain functional throughout the network. The energy consumption of these active nodes may be minimized further by rotating the selection of the 'best neighbour' considering the remaining battery power. The proposed scheme is better suitable for non-time-critical applications. This is another limitation of the proposed scheme.

In the future works, we would like to study, 1) the effect of variable sound velocity on the architecture formation, and 2) how the optimal placement of the surface sinks affects network performance, in our work.

ACKNOWLEDGEMENT

This work is partially supported by a grant from the Department of Electronics and Information Technology, Government of India, Grant No. 13(10)/2009-CC-BT, which the authors gratefully acknowledge.

REFERENCES

[1] Akyildiz, I. F., Pompili, D., and Melodia, T.: 'Underwater Acoustic Sensor Networks: Research Challenges'. Ad Hoc Networks, 2005, 2, (3) pp. 257–279

[2] Heidemann, J., Li, Y., Syed, A., Wills, J., and Ye, W.: 'Research Challenges and Applications for Underwater Sensor Networking'. Proc. of IEEE Wireless Communication and Networking Conference, 2006, pp. 228–235

[3] Cui, J.-H., Kong, J., Gerla, M., and Zhou, S.: Challenges: 'Building Scalable Mobile Underwater Wireless Sensor Networks for Aquatic Applications'. IEEE Network, 20, (3), pp. 12–18

[4] Kong, J., Cui, J.-H., Wu, D., and Gerla, M.: 'Building underwater Ad-hoc networks and Sensor Networks for Large Scale Real-time Aquatic Applications'. Proc. IEEE Military Communication Conference (MILCOM), 2005, pp. 1535–1541

[5] Sanchez, A., Blanc, S., Yuste, P., and Serrano, J. J.: 'A low cost and high efficient acoustic modem for Underwater Sensor Networks'. Proc. of IEEE OCEANS, 2011, pp. 1–10

[6] Shin, S., and Schulzrinne, H.: 'Measurement and Analysis of the VoIP Capacity in IEEE 802.11 WLAN'. IEEE Transactions on Mobile Computing, 2009, 8, (9), pp. 1265–1279

[7] Pompili, D., and Melodia, T.: 'An Architecture for Ocean Bottom UnderWater Acoustic Sensor Networks (UWASN)'. Proc. Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), Bodrum, Turkey, June 2004

[8] Pompili, D., Melodia, T., and Akyildiz, I. F.: 'Deployment Analysis in Underwater Acoustic Wireless Sensor Networks'. Proc. ACM WUWNet, September 2006, pp. 48–55

[9] Pompili, D., Melodia, T., and Akyildiz, I. F.: 'Three-dimensional and two-dimensional deployment analysis for underwater acoustic sensor networks'. Ad Hoc Networks, June 2009, 7, (4) pp. 778–790

[10] Cayirci, E., Tezcan, H., Dogan, Y., and Coskun, V.: 'Wireless sensor networks for underwater surveillance systems'. Ad Hoc Networks, 2006, 4, pp. 431–446

[11] Roy, S., Arabshahi, P., Rouseff, D., and Fox, W. L. J.: 'Wide Area Ocean Networks: Architecture and System Design Considerations', Proc. ACM WUWNet, 2006, pp. 25–32

[12] Seah, W. K.G., and Tan H.-X.: 'Multipath Virtual Sink Architecture for Underwater Sensor Networks'. Proc. IEEE OCEANS, 2006, pp. 1–6

[13] Climent, S., Capella, J. V., Meratnia, N., and Serrano, J. J.: 'Underwater Sensor Networks: A New Energy Efficient and Robust Architecture'. SENSORS, 2012, 12, pp. 704–731

[14] Wang, J., Li, D. Zhou, M. and Ghosal, D.: 'Data Collection with Multiple Mobile Actors in Underwater Sensor Networks'. Proc. IEEE Workshop on Delay/Disruption-Tolerant Mobile Networks (DTMN), June 2008, pp. 216–221

[15] Zhou, Z., Peng, Z., Cui, J.-H. Shi, Z., and Bagtzoglou, A. C.: 'Scalable Localization with Mobility Prediction for Underwater Sensor Networks'. IEEE Transaction on Mobile Computing, 2011, 10, (3), pp. 335–348

[16] NS-3 Simulator, http://www.nsnam.org/ (Accessed: Mar 29, 2013)

[17] Caruso, A., Paparella, F., Vieira, L. F. M., Erol, M., and Gerla, M.: 'The Meandering Current Mobility Model and its Impact on Underwater Mobile Sensor Networks'. Proc. IEEE INFOCOM, 2008, pp. 221–225

[18] Berkhovskikh, L., and Lysanov, Y.: 'Fundamentals of Ocean Acoustics'. (Springer, Germany, 1982)

[19] Yan, H., Shi, Z. J., and Cui J.-H.: 'DBR: Depth-Based Routing for Underwater Sensor Networks', Proc. IFIP-TC6 networking conference on Ad Hoc and sensor networks, wireless networks, next generation internet (NETWORKING), 2008, pp. 72–86