Design of an Address Assignment and Resolution Protocol for Underwater Networks

Rohit Agrawal
Department of EEE and E&I,
BITS Pilani-K. K. Birla Goa Campus.
Email: f2012267@goa.bits-pilani.ac.in

Mandar Chitre
Acoustic Research Laboratory,
Tropical Marine Science Institute,
National University of Singapore.
Email: mandar@arl.nus.edu.sg

Ahmed Mahmood
Acoustic Research Laboratory,
Tropical Marine Science Institute,
National University of Singapore.
Email: ahmed@arl.nus.edu.sg

***Abstract*—We propose a protocol that automates address assignment and resolution for nodes in an underwater network. The protocol resolves address conflicts and assigns a unique address to every node. It also informs each node about the addresses of the other nodes in the network. Using this protocol, any node can perform name resolution to find the address of a particular node. The protocol works in a distributed manner without dependence on any central node or database for address assignment and resolution. It is not only tested through simulations but also through deployment at sea.**

***Index Terms*—Underwater Networks, Address resolution, Address assignment, UnetStack.**

I. Introduction

Underwater communication has been used for many years, mainly for the purpose of oceanographic explorations. In the past few decades there have been major advances in this field and its applications now encompass diverse areas like (i) defense, (ii) oil explorations, (iii) pollution monitoring and (iv) disaster prevention [1] [2]. Some of the commonly cited challenges in underwater networks include low bandwidth, long propagation delay, half-duplex nature of the links, high packet loss, and time-variability [3] [4] [5]. To combat such challenges, it is important to develop and deploy optimized protocols that are specially designed for such environments.

Often nodes need to be deployed with minimum supervision. Such an application requires an address assignment and resolution protocol. The proposed protocol eases the burden on a user to assign addresses manually. Additionally, it also maintains a database of the neighboring nodes' addresses which may be used at any point in time to find the address of a particular node when sending data/control packets. It assigns non-conflicting addresses in the network, thereby ensuring the packets reach their intended destination. In addition to address assignment, the protocol also performs address resolution thereby enabling users to refer to each node by its name rather than a numerical address. Since centralized address resolution mechanisms are undesirable (as a single point of failure could disable the entire network), we propose a fully distributed approach to address assignment and resolution. To avoid collisions between packets, we introduce a random back-off time and retransmit the packets to increase the probability of receiving at least one packet from each of the nodes. An appropriate mathematical model has also been proposed by which the optimum values of both number of retransmits and maximum back-off time can be ascertained for a desired probability. The protocol is not only tested through simulations on UnetStack [6] but also through deployment at sea.

The rest of this paper is organized as follows: In section II, we discuss the design and requirements of the protocol. In section III, we analyze the simulation results obtained with the UnetStack simulator [6]. In section IV, we discuss certain optimization techniques and suggest mathematical models to prevent packet collisions in the protocol. We present experimental results in section V. Finally, we discuss some concluding remarks in section VI.

II. Design of the Protocol

A. Requirements

In comparison to terrestrial wireless networks that use radio-frequency (RF) signals, underwater acoustic networks (UANs) have large propagation delays and lower data rates. Due to these characteristics, network protocols designed for terrestrial networks cannot be directly used in UANs [7]. The following requirements are taken into consideration while designing our proposed protocol:

- There should not be any conflicts in address assignment, i.e. at any point in time there should not be two nodes in the network with the same address.
- The protocol should perform name to address resolution in addition to address assignment.
- It should be distributed and not have any centralized or hierarchical dependence on any node or a set of nodes.
- It should take into account the large propagation delays in the network and the half duplex nature of communication links.
- The packets sizes used in the protocol should be small due to the lower data rates in underwater modems compared to terrestrial wireless networks.
- The protocol is designed for use in small networks (typically less than 100 nodes).

B. Address Assignment Protocol

Consider a network with n nodes where $n-1$ nodes have addresses assigned and the n^{th} node has just joined and

TABLE I
PACKET SUMMARY.

Packet Name	Abbreviation	Size (bytes)	Contents
Address Assignment Protocol			
Initial Hash Packet	IHP	2	Msg Id, Hash
Address Table Packet	ATP	$x+2$	Msg ID, Node address & x neighbour addresses
Conflict Notification Packet	CNP	2	Msg Id, Suggestion
New Hash Packet	NHP	2	Msg Id, New Hash
Final Address Packet	FAP	2	Msg Id, Final address
Address Resolution Protocol			
Initial Name Packet	INP	2	Msg Id, Node name
Resolution Failure Packet	RFP	2	Msg Id
Final Name Packet	FNP	variable	Msg Id, Node name
Resolved Address Packet	RAP	2	Msg Id, Final address

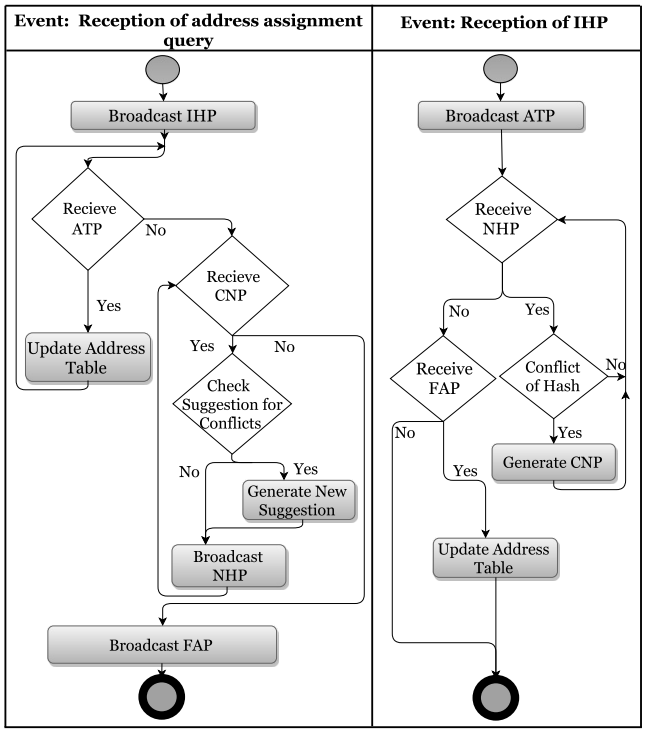

Fig. 1. Flow chart of the address assignment protocol.

needs address assignment. The address assignment begins by generating a hash based on the name of the node using the Fletcher check sum algorithm [8]. A hashing algorithm is used to convert the node name of variable length into a numeric address. This hash is broadcasted as the Initial Hash Packet (IHP) to all the nodes in the network to check for conflicts. The $n-1$ neighbor nodes check the received hash with their own addresses and within the entries of their address table for possible conflicts. If a conflict occurs, the corresponding node replies with a Conflict Notification Packet (CNP) indicating a conflict. The CNP also contains a suggestion which is the next numerically incremented hash that is not present in its own address table. For eg., if the conflicting hash is 74 and 75 is not present in the node's address table then 75 is broadcasted as a suggestion via the CNP. All $n-1$ nodes also send an Address Table Packet (ATP). This packet consists of their own address and x addresses from their own address table. Upon receiving the ATPs, each node updates its address table. This ensures that all nodes have the addresses of their immediate neighbors as well as some other nodes in the network. If the n^{th} node receives a CNP, it checks the received suggestion for conflicts within the entries of its own address table. If it faces a conflict, it generates a new suggestion that is not present in its own address table. The suggestion along with the message ID is then broadcasted as a New Hash Packet (NHP) to the neighboring nodes in the network. The nodes in the network check this new hash for conflicts and if it conflicts they reply back with a CNP again. This process continues until the conflict has been resolved. Upon resolution of the conflict, the node is assigned the nonconflicting address which is broadcasted as the Final Address Packet (FAP) to the neighboring nodes which then update their address table with the node's address.

Each packet that is sent out in this protocol has a unique message ID as its first byte. This helps in differentiating between packets and taking appropriate actions. Details of the packets are shown in Table I. To illustrate the message exchanges that happen as part of the address assignment protocol, a flow chart is presented in Fig. 1. The various cases of a nonconflicting hash and a conflicting hash are also explained with sequence diagrams presented in Fig. 2 and Fig. 3.

C. Address Resolution Protocol

When a node receives a node name for resolution via the address resolution query, it first checks its own node name and address table for the particular node name. In case the entry is missing, the protocol generates a hash corresponding to the node name and sends this name for the resolution to the node in the network having the corresponding hash as its address. There is a high chance of finding the entry in the latter's table because during its own address assignment it would have generated the same hash and received ATPs. In case it could not assign itself that hash, it may have information about the conflicting node having this hash as its address in its address table. If the entry is found, the node replies back with a Resolved Address Packet (RAP) thereby completing address resolution. If the entry is not present, it replies back with a Resolution Failure Packet (RFP) to the node which received the address resolution query. On receiving a RFP, the node broadcasts a Final Name Packet (FNP). Any node which has

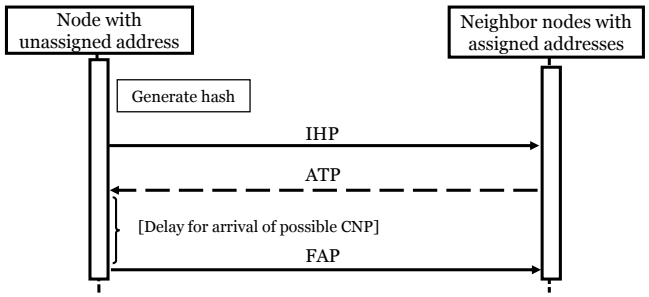

Fig. 2. Sequence diagram of the address assignment protocol for a nonconflicting hash.

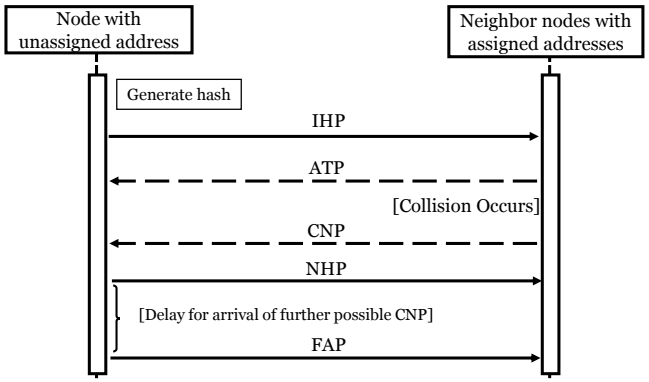

Fig. 3. Sequence diagram of the address assignment protocol for a conflicting hash.

the particular entry in its table replies back with a RAP thereby completing address resolution.

Similar to address assignment, each packet in the address resolution protocol also has a unique message ID. Details of the packets are shown in Table I. To illustrate the message exchanges that take place as part of the address resolution protocol, a flow chart is presented in Fig. 4.

III. Simulation Results

The address assignment protocol and the address resolution protocol are simulated on the UnetStack Simulator [6]. In UnetStack, the stack consists of a collection of software agents that provide well-defined services. The address assignment protocol and the address resolution protocol are developed as agents in UnetStack. These agents are loaded on all the nodes in the network and the simulations are carried out.

A. Address Assignment Protocol

The address assignment protocol is simulated for an 8 and a 64 node network. Each ATP contains addresses of 4 neighbor nodes and hence each ATP is of 6 bytes. We discuss the simulation results in detail in the following subsections.

1) 8 Node Network: The protocol is simulated on a network with one unassigned address node and seven neighbor nodes which have addresses assigned. The network geometry is as shown in Fig. 5 with the node with an unassigned address at the origin. The node names are stated in quotes. For

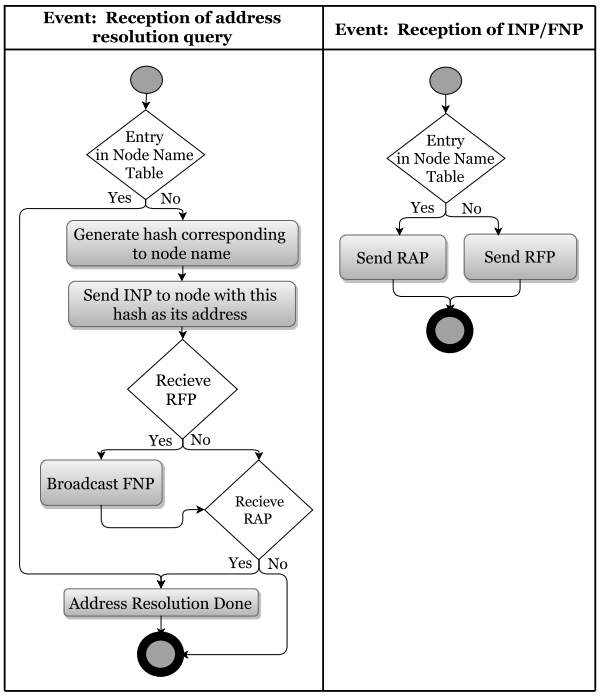

Fig. 4. Flow chart of the address resolution protocol.

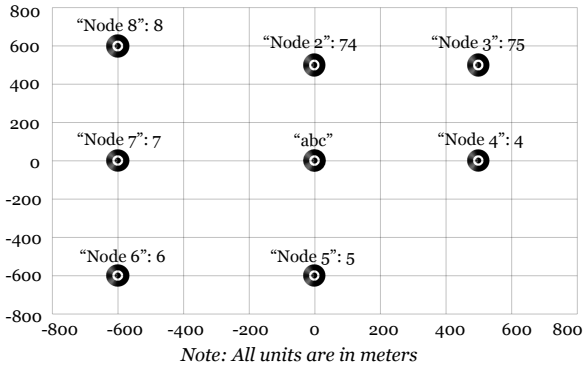

Fig. 5. Network geometry for address assignment protocol.

the purpose of simulation and to depict a double conflict scenario i.e, a conflicting hash being resolved with the second suggestion received, address tables of all nodes are kept blank except 'Node 3' which has an entry of 76. Node 'abc', which is at the origin does not store ATPs received from 'Node 3'. This prevents the conflict being resolved with the first suggestion received via the CNP. If ATPs from 'Node 3' are stored, the first NHP broadcasted will have nonconflicting address (77) thereby completing address assignment.

The node 'abc' begins address assignment by sending an address assignment query to the address assignment agent. The name 'abc' corresponds to a hash of 74 being generated. On receiving this hash the other 7 nodes send their addresses and 'Node 2' also sends a CNP containing the suggestion 75. Since 75 is not present in the address table of 'abc', this suggestion is now broadcasted as the NHP. Upon reception, 'Node 3'

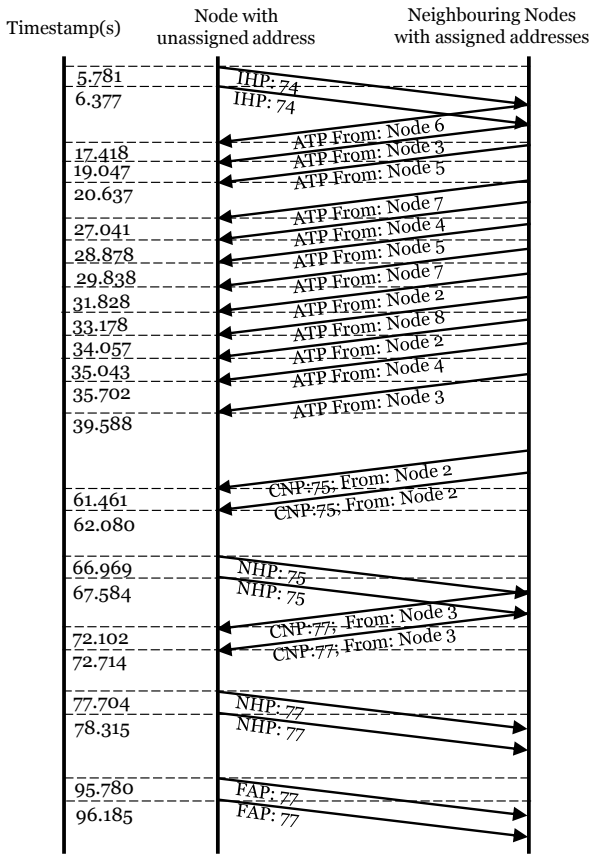

Fig. 6. Timeline of address assignment protocol.

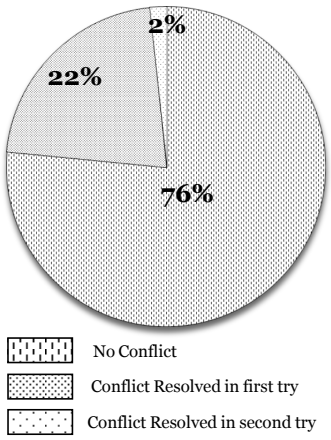

Fig. 7. Percentage of nodes facing no conflicts and conflicts.

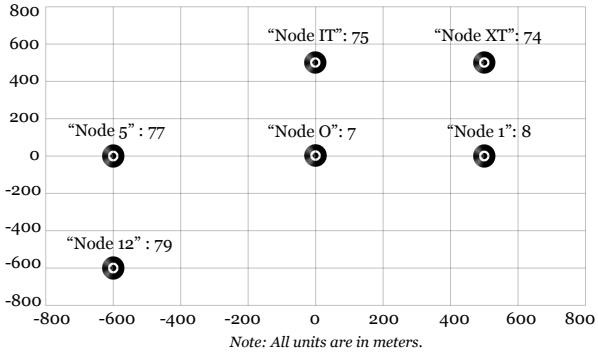

Fig. 8. Network geometry for address resolution protocol.

sends a CNP containing the suggestion 77. This suggestion is again broadcasted as a NHP by node 'abc' to check for further conflicts. Since this hash does not conflict with any address in the network, it is assigned to node 'abc'. A visualization of transmission and reception of packets in the form of a timeline is shown in Fig. 6.

2) 64 Node Network: To ensure scalability of the protocol, we also simulate a 64 node network. All 64 nodes are randomly placed in a circle of radius of 1 km. Each node is assigned a random node name. The network grows from a 1 node network gradually to a 64 node network where nodes keep joining one after the other and perform address assignment sequentially. Since initially there are lesser nodes in the network, there are lesser conflicts. However, as the network grows (since most hashes are already taken), conflicts start occurring and they are resolved. Apart from testing scalability, the simulation also provides information of the number of tries needed for address assignment. With features like a hashing algorithm, address tables and suggestions provided on conflicts incorporated in the protocol, the objective is to reduce the number of conflicts that could occur each time a hash is broadcasted, thereby reducing the overall time required for address assignment.

Of the 64 nodes in the network, 49 nodes generate hashes which do not collide with existing nodes. 15 nodes face conflicts with the hash generated, 14 of which are resolved with the first suggestion received while only 1 node has to attempt a second retry for address assignment. A graphical illustration of the same in the form of a pie chart is shown in Fig. 7.

TABLE II
SIMULATION RESULTS FOR ADDRESS RESOLUTION.

Node Name Query	Number of Tries	Time taken (s)
Node 1	1	0.110
Node XT	2	3.870
Node IT	3	8.870

B. Address Resolution Protocol

The address resolution protocol is simulated on the network with node names in quotes and node addresses as shown in Fig. 8. The simulation provides information about the following three different possibilities of address resolution:

(a) The node name is present in the node's own address table.
(b) The node name is present in the address table of the node having the corresponding hash as its address.

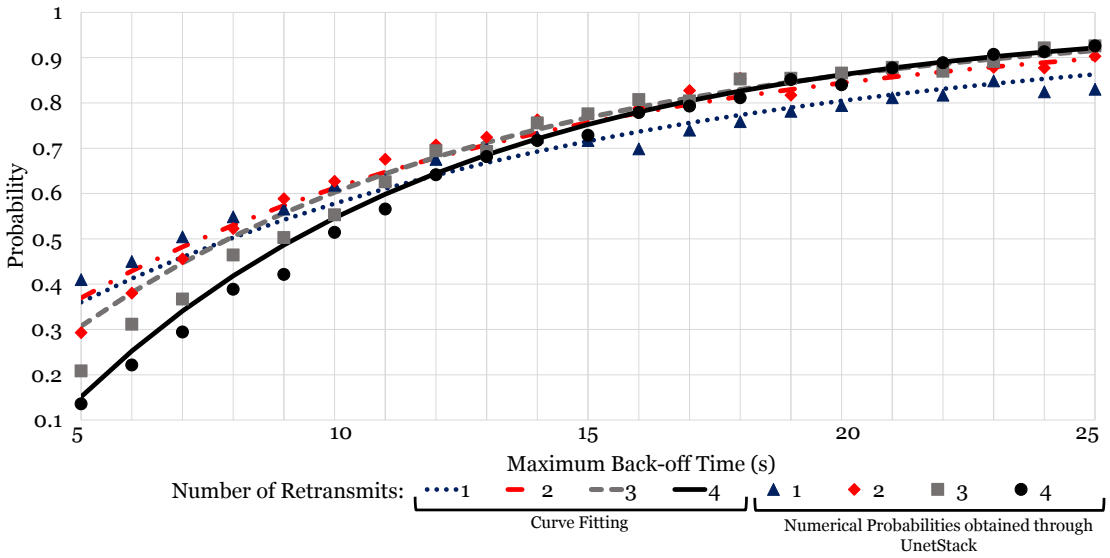

Fig. 9. Probability Curves obtained from simulation and through curve fitting.

(c) The third possibility is when both the previous attempts have failed and a FNP is broadcasted. Any node which has the particular entry in its table replies back with a RAP.

For the purpose of simulation and to depict all 3 scenarios of address resolution, mentioned above address tables of all nodes are kept blank except 'Node O' which has entry for 'Node 1'. Similar to address assignment, the address resolution protocol is simulated as an agent loaded on all the nodes in the UnetStack.

Three address resolution queries are sent to to 'Node O'. The purpose of each query is to test the three different possibilities of address resolution. For the first possibility of entry being in the node's own address table, a query for 'Node 1' is sent. For the second possibility of the entry being known by node with corresponding hash as its address, a query for node name 'Node XT' is sent. Since 'Node XT' corresponds to the hash 74, the INP is sent directly to this node to which it replies with a RAP. The final possibility is when both the previous attempts have failed. In this case the FNP is broadcasted to all nodes. To validate this, we send a query for node name 'Node IT' which corresponds to a hash of 74. Since the node with address 74 ('Node XT') has a blank address table it replies back with a RFP. Following this, a FNP is broadcasted, to which 'Node IT' replies with its own address as the RAP. A summarized table containing the details of the time taken for the three different queries is shown in Table II.

IV. Reducing Packet Collisions

As stated previously, our protocol updates the address table of a new node in the network by receiving ATPs from all neighboring nodes. If there are n such nodes, then packet collisions are bound to occur especially as n increases. To reduce these collisions, a random back-off time is introduced. Each packet is also retransmitted to ensure *at least one* ATP is received from each neighbor node.

It is prudent to find suitable values of the maximum back-off time and number of retransmits to minimize packet collision. We note that if both parameters are increased, packet collisions may be reduced. However, doing so would be accompanied by long delays to achieve address assignment. We simulate a network of 9 nodes on UnetStack where 8 nodes send ATPs to 1 node. In Fig. 9, we show the probability (data points) of receiving at least one ATP from each of the 8 neighbor nodes against the maximum back-off time for retransmissions within $\{1, 2, 3, 4\}$. The maximum back-off time is measured in seconds and constrained to the set $\{5, \ldots, 25\}$. For each back-off time and retransmit pair, 100 simulations are run on the UnetStack and the probability of receiving at least one ATP from each node is computed.

Though one can undertake more rigorous simulations to evaluate probabilities for intermediate back-off times and higher number of retransmits, a more useful method would predict these probabilities via an appropriate mathematical model. This is done next.

A. Curve Fitting

To fit the data points in Fig. 9 with a suitable mathematical model, we first note that the latter should output values in the interval $[0, 1]$ and take as its argument a function of the back-off time (x_1) and number of retransmits (x_2). Let us assume N data points. Then the i^{th} coordinate pair may be represented as $(x_1^{(i)}, x_2^{(i)}, y^{(i)})\ \forall\ i \in \{1, 2, \ldots, N\}$, where $x_1^{(i)}$ is the back-off time, $x_2^{(i)}$ is the number of retransmits and $y^{(i)}$ is corresponding probability.

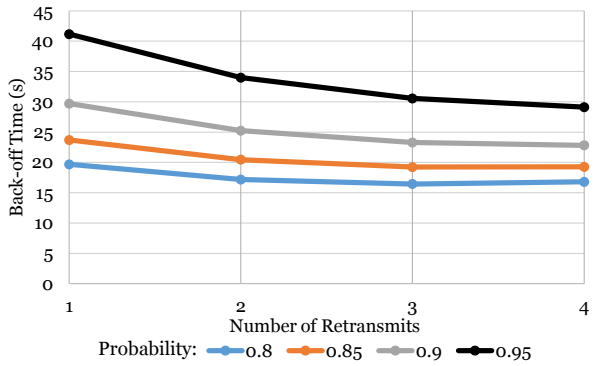

Fig. 10. Back-off Time vs Number of Retransmits for various desired probabilities.

For our fitting problem, we assume the polynomial feature vector $\mathbf{x} = [x_1, x_2, x_1^2, x_2^2, x_1x_2]^\mathsf{T}$. Similarly, the i^{th} data vector is $\mathbf{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, x_1^{2(i)}, x_2^{2(i)}, x_1^{(i)}x_2^{(i)}]^\mathsf{T}$. Defining

$$\boldsymbol{\mu}_\mathbf{x} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)} \text{ and}$$

$$\boldsymbol{\sigma}_\mathbf{x}^2 = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \boldsymbol{\mu}_\mathbf{x})$$

as the data mean and variance vectors respectively, we have the normalized feature vector

$$\bar{\mathbf{x}} = \begin{bmatrix} 1 \\ (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})/\boldsymbol{\sigma}_\mathbf{x} \end{bmatrix}. \tag{1}$$

We employ the mathematical model:

$$h_{\boldsymbol{\theta}}(\bar{\mathbf{x}}) = 1 - e^{-\boldsymbol{\theta}^\mathsf{T}\bar{\mathbf{x}}}, \tag{2}$$

where $\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_5]^\mathsf{T}$ are the parameters that are to be estimated.

By adopting the aforementioned framework, the problem may be reduced to that of linear regression. On taking the logarithm of (2) and simplifying, we obtain

$$\boldsymbol{\theta}^\mathsf{T}\bar{\mathbf{x}} = \underbrace{\log(1 - h_{\boldsymbol{\theta}}(\mathbf{x}))}_{\bar{h}_{\boldsymbol{\theta}}(\bar{\mathbf{x}})}.$$

Let $\bar{y}^{(i)} = \log(1 - y^{(i)})$, then the mean squared error (MSE) between the *transformed* hypothesis $\bar{h}_{\boldsymbol{\theta}}(\bar{\mathbf{x}})$ and data probabilities is

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(\bar{y}^{(i)} - \bar{h}_{\boldsymbol{\theta}}(\bar{\mathbf{x}}^{(i)}))^2$$

$$= \frac{1}{N}\sum_{i=1}^{N}(\bar{y}^{(i)} - \boldsymbol{\theta}^\mathsf{T}\bar{\mathbf{x}}^{(i)})^2, \tag{3}$$

where $\bar{\mathbf{x}}^{(i)}$ is the i^{th} normalized data vector corresponding to (1), i.e.,

$$\bar{\mathbf{x}}^{(i)} = \begin{bmatrix} 1 \\ (\mathbf{x}^{(i)} - \boldsymbol{\mu}_\mathbf{x})/\boldsymbol{\sigma}_\mathbf{x} \end{bmatrix}. \tag{4}$$

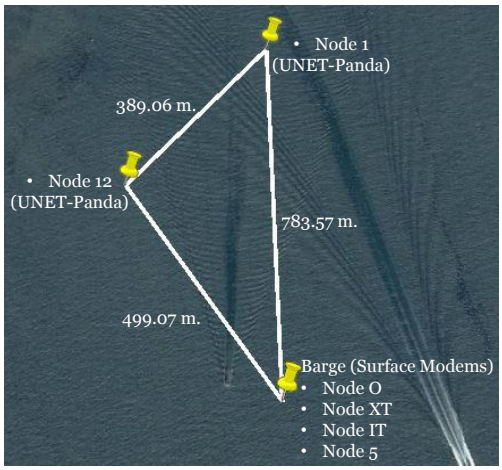

Fig. 11. Node deployment as a part of UNET-2015.

Minimizing (3) is straightforward and may be done via the normal equations [9]

$$\hat{\boldsymbol{\theta}} = (\bar{\mathbf{X}}^\mathsf{T}\bar{\mathbf{X}})^{-1}\bar{\mathbf{X}}^\mathsf{T}\bar{\boldsymbol{y}}, \tag{5}$$

where

$$\bar{\mathbf{X}}^\mathsf{T} = [\bar{\mathbf{x}}^{(1)}, \bar{\mathbf{x}}^{(2)}, \ldots, \bar{\mathbf{x}}^{(N)}] \text{ and} \tag{6}$$

$$\bar{\mathbf{y}} = [\bar{y}^{(1)}, \bar{y}^{(2)}, \ldots, \bar{y}^{(N)}]. \tag{7}$$

For the data points in Fig. 9, $N = 100$. The fitting resulted in

$$\hat{\boldsymbol{\theta}} = \begin{bmatrix} 1.16 \\ 0.54 \\ 0.12 \\ -0.07 \\ -0.31 \\ 0.34 \end{bmatrix}, \quad \boldsymbol{\mu}_\mathbf{x} = \begin{bmatrix} 13.0 \\ 2.5 \\ 221 \\ 7.5 \\ 32.5 \end{bmatrix} \text{ and } \boldsymbol{\sigma}_\mathbf{x} = \begin{bmatrix} 7.24 \\ 1.12 \\ 194.11 \\ 5.70 \\ 24.64 \end{bmatrix}.$$

In Fig. 9, we have also plotted the fitted curves and have color-coded them to their corresponding data points. Clearly, the mathematical model in (2) tracks the data points well. The probability of receiving ATPs as a function of the maximum back-off time and number of retransmits is given by

$$P(x_1, x_2) = 1 - e^{-\boldsymbol{\theta}^\mathsf{T}\bar{\mathbf{x}}}. \tag{8}$$

Eq. (8) now may be used to find (x_1, x_2) pairs for a desired probability. We evaluate the two parameters for four sample probabilities in Fig. 10. The corresponding back-off time and number of retransmits can then be ascertained. For our use, we set the back-off time to 19s and the number of retransmits to 2 to achieve $P(19, 2) = 0.85$ for our protocol.

V. Experimental Results

The protocol was tested in the waters of Singapore near Selat Puah as a part of the UNET-2015 experiments. The deployed network consisted of 6 nodes: 2 bottom mounted UNET-Panda [10] nodes, and 4 surface modems deployed from a barge. The locations of the nodes are shown in Fig. 11. The exact depth of the modem varied depending on the

prevailing currents and each of them were about 2 m off the seabed. The water depth in the area is between 7 and 20 m, typically shallow close to the islands and deeper in the middle of the channel. The acoustic modem installed in the UNET-Panda is the ARL UNET-2 modem [11]. It operates in the 18-36 kHz frequency band and has a maximum range of about 2.5 km.

A. Address Assignment Protocol

Each node in the network performed address assignment sequentially starting from 'Node O' to 'Node 12' as illustrated in Table III. 'Node O' and 'Node XT' generated hashes which did not collide and hence were able to assign addresses as the corresponding hash. 'Node 1' and 'Node IT' faced conflicts which were resolved with the first suggestion received. For the purpose of creating a double conflict scenario for 'Node 5' and 'Node 12', 'Node O' contained only 2 entries in its address table which were 76 and 78. ATPs sent by 'Node O' were neither stored by 'Node 5' nor 'Node 12'. This forced both the nodes to attempt a second retry for address assignment. The duration of the experiment starting from the address assignment query sent by 'Node O' to FAP being sent by 'Node 12' was 960 seconds.

TABLE III
EXPERIMENTAL RESULTS FOR ADDRESS ASSIGNMENT PROTOCOL.

Node Name	Hash Generated	Address Assigned
Node O	7	7
Node 1	7	8
Node XT	74	74
Node IT	74	75
Node 5	75	77
Node 12	77	79

B. Address Resolution Protocol

The address resolution protocol was tested on this established network, with all nodes having addresses assigned. Testing of the address resolution protocol was performed in a similar manner as the simulations in UnetStack. We tested the three different possibilities of address resolution by sending three possible queries. All queries were sent to 'Node O'. For the purpose of testing, all address tables were kept blank except for 'Node O' which had an entry for 'Node 1'. Hence the query for 'Node 1' took one attempt. The second query for 'Node XT' which corresponds to a hash of 74 took 2 attempts as it was not present in Node O's address table. The third query for 'Node IT' which also corresponds to a hash of 74 took three attempts since it was neither present in Node O's address table nor in the node which had an address of 74 which was 'Node XT'. The three different queries and the respective timings are shown in Table IV.

TABLE IV
EXPERIMENTAL RESULTS FOR ADDRESS RESOLUTION PROTOCOL.

Node name query	Number of attempts	Time taken (s)
Node 1	1	0.028
Node XT	2	3.673
Node IT	3	12.739

VI. CONCLUSION

We proposed and analyzed the address assignment and resolution protocol in detail. Not only does it perform well in simulation but also when deployed at the sea. The scalability of the protocol was also been demonstrated. A mathematical model reducing packet collisions has also been suggested. Future work may incorporate a routing protocol in the agent.

REFERENCES

[1] S. Rajasegarar, J. Gubbi, O. Bondarenko, S. Kininmonth, S. Marusic, S. Bainbridge, I. Atkinson, and M. Palaniswami, "Sensor network implementation challenges in the great barrier reef marine environment," in *Proceedings of the ICT-MobileSummit 2008 Conference*, 2008.

[2] K. Liu, Z. Yang, M. Li, Z. Guo, Y. Guo, F. Hong, X. Yang, Y. He, Y. Feng, and Y. Liu, "Oceansense: Monitoring the sea with wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 14, no. 2, pp. 7–9, 2010.

[3] I. F. Akyildiz, D. Pompili, and T. Melodia, "State-of-the-art in protocol research for underwater acoustic sensor networks," in *Proceedings of the 1st ACM international workshop on Underwater networks*. ACM, 2006, pp. 7–16.

[4] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, pp. 23–33, 2007.

[5] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater acoustic communications and networking: Recent advances and future challenges," *Marine technology society journal*, vol. 42, no. 1, pp. 103–116, 2008.

[6] M. Chitre, R. Bhatnagar, and W.-S. Soh, "Unetstack: an agent-based software stack and simulator for underwater networks," in *Oceans-St. John's, 2014*. IEEE, 2014, pp. 1–10.

[7] I. F. Akyildiz, D. Pompili, and T. Melodia, "Underwater acoustic sensor networks: research challenges," *Ad hoc networks*, vol. 3, no. 3, pp. 257–279, 2005.

[8] J. Fletcher, "An arithmetic checksum for serial transmissions," *IEEE Transactions on Communications*, pp. 247–252, 1982.

[9] S. Boyd and L. Vandenberghe, *Convex Optimization*, ser. Berichte über verteilte messysteme. Cambridge University Press, 2004. [Online]. Available: https://books.google.com.sg/books?id=mYm0bLd3fcoC

[10] M. Chitre, I. Topor, R. Bhatnagar, and V. Pallayil, "Variability in link performance of an underwater acoustic network," in *OCEANS-Bergen, 2013 MTS/IEEE*. IEEE, 2013, pp. 1–7.

[11] M. Chitre, I. Topor, and T.-B. Koay, "The unet-2 modem — an extensible tool for underwater networking research," in *OCEANS, 2012-Yeosu*. IEEE, 2012, pp. 1–7.