COPYRIGHT NOTICE

Federation UNIVERSITY • AUSTRALIA

FedUni ResearchOnline
http://researchonline.federation.edu.au

This is the peer-reviewed version of the following article:

Nowsheen, N., Karmakar, G., Kamruzzaman, J. (2014) An adaptive approach to opportunistic data forwarding in underwater acoustic sensor networks. 2014 13th International Symposium on Network Computing and Applications, NCA 2014*, 229-236.*

Which has been published in final form at:
http://doi.org/10.1109/NCA.2014.41

Copyright © 2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

An Adaptive Approach to Opportunistic Data Forwarding in Underwater Acoustic Sensor Networks

Nusrat Nowsheen
Faculty of Information Technology
Monash University
Churchill, VIC 3842, Australia
Email: nusrat.nowsheen@monash.edu

Gour Karmakar and Joarder Kamruzzaman
Faculty of Science
Federation University Australia
Churchill, VIC 3842, Australia
Email: {gour.karmakar, joarder.kamruzzaman}@federation.edu.au

***Abstract*—Reliable data transfer for underwater acoustic sensor networks (UASNs) is a major research challenge in applications such as pollution monitoring, oceanic data collection, and surveillance due to the long propagation delay and high error rate of the acoustic channel. To address this issue, an opportunistic data forwarding protocol [1] was proposed which achieves high packet delivery success ratio with less routing overhead and energy consumption by selecting the next hop forwarder among a set of candidates based on its link reliability and data transfer reachability. However, the protocol relies on fixed data hold time approach, i.e., each node holds data packets for a fixed amount of time before a forwarder discovery process is initiated. Depending on the value of the fixed hold time and deployment contextual scenario, this may incur large end-to-end delay. Moreover, lack of consideration of network condition in hold time limits its performance. In this paper, we propose an adaptive technique to improve its performance. The adaptive approach calculates data hold time at each node dynamically considering a number of 'node and network' metrics including current buffer occupancy, delay experienced by stored data packets, arrival and service rate, neighbors' data transmissions and reachability. Simulation results show that compared with fixed hold time approach, our adaptive technique reduces end-to-end delay significantly, achieves considerably higher data delivery and less energy consumption per successful packet delivery.**

***Keywords*—underwater acoustic sensor networks; delay tolerant networks; opportunistic data forwarding; hold time;**

I. Introduction

Recently, considerable research efforts have been put to Underwater Acoustic Sensor Networks (UASNs) [2], [3], [4], [5] to enable communications among a number of sensor nodes and vehicles deployed at different levels of the ocean. The network enables a wide range of applications, such as monitoring ocean environment, oceanographic data collection, disaster prevention, navigation and surveillance, etc. [2]. Unlike terrestrial wireless networks that mainly rely on RF communications, UASNs utilize acoustic communications. The reason to use acoustic is that RF signals attenuate quickly under water due to high absorption and thus can merely propagate a few meters in such environment. However, the unique characteristics of the underwater acoustic channel poses new research challenges in the design of UASNs, such as, long propagation delay due to the low speed of acoustic signals under water (apprx. 1500 m/s), high bit error rate, limited bandwidth and high transmission power [2], [5]. Due to the above unique characteristics, data forwarding protocols for terrestrial radio networks are inefficient for underwater environment [2], [5].

In the existing literature [6], [7], UASN has been viewed as a Delay/Disruption Tolerant Network (DTN). Traditional DTN solutions exploit node mobility as a significant factor to support opportunistic forwarding when a next hop is found in the communication range. As a result, DTN routing is based on *'store-carry-forward'* paradigm where participating nodes store and carry data destined for another node until i) the final destination is encountered or ii) a suitable forwarder is found which can '$store\ and\ carry$' those data. However, the traditional DTN solutions for data forwarding are not suitable for UASNs because underwater sensor nodes are mostly quasi-stationary (anchored to the ocean bottom with a cable/wire). Nodes in this network do not suffer from communication disruption only due to mobility as is the case for traditional DTNs. Furthermore, underwater communication is prone to interferences, disruption and unpredictable delay due to the harshness of the environment and slow nature of the acoustic medium. To support delay tolerant applications in underwater networks, a modified form of DTN solution can be used to improve data transfer reliability in such a dynamic environment. Therefore, we have adopted *'accumulate-and-forward'* technique to improve channel efficiency where message overhead is reduced by exchanging one control packet (*Request-Reply*) for a bunch of data packets as described in [1].

In our previous work [1], an opportunistic data forwarding solution is devised to improve reliability in data transfer in UASNs. Our data forwarding solution achieves high successful data delivery ratio by selecting the next hop forwarder based on its link reliability and data transfer reachability. For each node, the protocol accumulates a train of data packets for a fixed amount of time (called 'hold time') and performs a forwarder discovery process after it expires. If a forwarder is found, the protocol transmits accumulated data back-to-back in one burst to it. However, accumulation of data packets for a fixed amount of time without considering the local network conditions can cause the packets in the buffer to experience large delay at each hop before being forwarded to the next hop. This waiting delay in turn increases the end-to-end delay that a packet suffers during its transmission from source to the gateway (GW) node. Failure of forwarder discovery within a certain time limit further increases the waiting delay for a fixed amount of time in this protocol. Therefore, the existing solution sacrifices latency in order to achieve higher data delivery ratio.

However, it is observed that the performance of our opportunistic data forwarding solution can be further improved if each node in the network calculates their hold time according to their local network statistics. The local network statistics (e.g., packet arrival and effective data transfer rate, local buffer delay and occupancy, etc.) vary from node to node. Each node in the network does not need to suffer from the same amount of waiting delay if their local network statistics are different. This motivates us to re-explore our existing solution which uses a fixed hold time at each node. It makes much more sense if the protocol can adapt to the current network condition and adjusts its data storing behavior accordingly rather than always accumulating and storing data packets for a fixed amount of time, at each hop.

There are already several holding time estimation techniques [8], [9], [10], [11], [12] available in the literature to improve the performance of data forwarding techniques in UASNs. However, they are designed to reduce the number of redundant transmissions which occur due to broadcasting activities of their data forwarding protocols. Therefore, these existing hold time estimation techniques are not suitable for improving the performance of our unicast data forwarding protocol [1] as they were developed specifically for data broadcasting and require either depth or direct location information of a sensor node. Moreover, none of these protocols consider local network conditions, which are important to enhance data transfer reliability, for calculating hold time.

Motivated by this fact, we propose an adaptive hold time estimation technique to our opportunistic data forwarding solution. The protocol initiates forwarder discovery mechanism dynamically according to its calculated hold time based on some local network specific parameters to reduce end-to-end delay. To further improve data delivery, the proposed technique attempts to reduce collisions by exploiting the one hop forwarder discovery technique. In this regard, control messages exchanged during forwarder discovery are used to adjust the hold time at each node according to neighbors' data transmission information. The main contributions of this paper are:

- An adaptive technique to an opportunistic data forwarding protocol by adjusting data hold time dynamically considering local network statistics is developed. Factors incorporated include delay experienced by each node's local buffer, service time of all stored packets, buffer fill-up time, packet arrival and effective data transfer capacity of an outgoing link,
- Hold time is further adjusted considering neighbors' data transmissions to reduce collisions in the network, and
- Performance analysis of the proposed technique is conducted through the simulation models developed in ns2 which considers fully acoustic environment by modelling the channel in the physical layer [13]. The simulation results exhibit the improvement of packet delivery, end-to-end delay and energy consumption per successful packet delivery, due to adaptive hold time.

The remainder of this paper is organized as follows. Section II briefly reviews existing works. Section III provides an overview of our previous forwarding approach [1] using fixed hold time and highlights our motivation towards adaptive approach using variable hold time to an opportunistic data forwarding solution. Section IV shows the simulation results in ns2. Finally Section V concludes the paper.

II. Related Works

In this section, we review some notable works on DTN and underwater routing protocols to demonstrate how their data forwarding solutions differ from our work.

A. Data Forwarding for DTN Networks

Several protocols have been proposed in DTN based on various assumptions regarding connectivity and availability of environmental knowledge. Dissemination based routing [14], [15] relies on replicating data packets all over the network. These protocols hold data packets until a forwarder or destination is found into contact through node mobility. These techniques ensure optimal message delivery rate and minimizes the delay experienced by each message. However, they are very resource intensive and consume plenty of network bandwidth and storage capacity and are thus very energy inefficient. Utility based routing [16], [17] limits the number of copies of a message that are spread over the network and selects an appropriate relay node based on some utility function which estimates the usefulness of a host as a next hop. They hold data packets until a suitable forwarder (higher utility value) is found in the communication range. Utility based techniques optimize the overall bandwidth usage and energy consumption of the participating nodes. However, they have higher computation cost than dissemination based techniques as nodes need to maintain a state that keeps track of the utility values associated to all other nodes in the networks. The cost of updating the state at each node also increases the overall protocol overhead. However, the above protocols rely on node mobility for data forwarding and therefore, not suitable for harsh, error-prone underwater environment with quasi-stationary architecture.

B. Data Forwarding for UASNs

Several routing protocols have been proposed in UASNs. In this section, we present some related protocols in UW that use the concept of hold time for packet forwarding.

1) Depth-based Forwarding: Depth-based Routing (DBR) [8] is an underwater routing protocol which forwards data packets greedily from higher to lower depth node in a broadcast fashion until data packets reach the sink. DBR introduces holding time in their technique to reduce redundant packet transmissions. When a node receives a data packet, it first holds the packet for a certain amount of time. Each node schedules packet sending time based on the calculated holding time. At each node, the holding time is calculated for each individual received packet based on the depth difference between the previous hop and the depth of the current node. However, DBR cannot handle void zone problem, thus fails to achieve high packet delivery rate in sparse areas. Moreover, sensor nodes having similar depth have small differences in their holding time which is not long enough for overhearing. Hence, redundant packet transmissions are still unavoidable in this technique. Energy Efficient Depth-Based Routing (EEDBR)

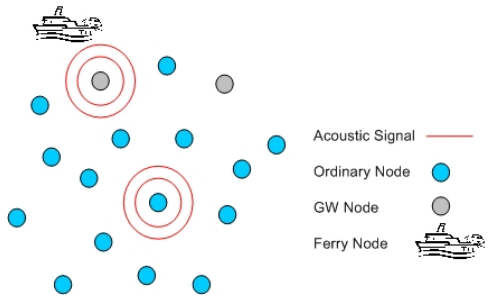

Fig. 1. Network Model

[9] is another depth-based routing protocol which selects neighboring forwarder based on the depth and residual energy. Like DBR, sensor nodes in this protocol hold the data packets for a certain amount of time before forwarding to reduce redundant packet transmission. The holding time is calculated based on the residual energy of sensor nodes. A node having high residual energy has a shorter holding time compared to the nodes having low energy. This allows nodes with high residual energy to forward data packets while restrict nodes with low energy to suppress packet transmission upon overhearing the transmission of the same packet. However, the protocol relies on a knowledge acquisition phase to update sensor nodes about more recent neighboring nodes and their depth and residual energy. This incurs large message overhead in EEDBR. Similar to DBR, Energy-based DBR (EDBR) [10] forwards data packets based on depth information. The protocol also uses holding time to schedule packet forwarding. Unlike DBR which employs only depth information to calculate holding time, EDBR utilizes both the depth information and residual energy of participating nodes to calculate holding time. However, the protocol has the same disadvantage as that of DBR.

2) Location-based Forwarding: Relative Distance Based Forwarding (RDBF) protocol [11] uses a 'fitness factor' to measure the appropriateness of a next hop to be a forwarder. It limits the scope of the candidate forwarders and finds the beneficial relays to forward data packets. A node with high 'fitness factor' is near enough to the sink node and suitable for relaying data packets. The protocol uses broadcast technique to forward data packets. When a node receives a data packet, it calculates the fitness factor from the forwarder's and its distance to the sink. It then calculates data holding time based on the 'fitness factor'. The node having a high fitness factor (closer to the sink) will hold packets for a short amount of time. Thus, RDBF involves a small number of nodes in forwarding to reduce the energy consumption in the network. However, the protocol requires each source node to know its own 3D location as well as the location of sink node which requires sink node to notify its location to the network termly. In Energy-efficient Routing Protocol based on Physical distance and Residual energy (ERP^{2}R) [12], each forwarder calculates holding time (the waiting time till a packet transmission) based on residual energy. A forwarder suppresses its transmission upon overhearing a transmission from a node having higher residual energy. The node having more energy has shorter holding time than the node with low energy. The protocol also uses a priority value to avoid the nodes having small differences in their residual energy to have similar holding time. ERP^{2}R also relies on a knowledge acquisition phase to update information about neighbor's physical distance to the sink node and their residual energy. This incurs large message overhead. Furthermore, the physical distance to the sink is estimated through Time of Arrival (ToA) which requires clock synchronization among sensor nodes.

III. Adaptive Approach to Opportunistic Data Forwarding

This section begins with a brief discussion of our existing technique [1] based on fixed hold time. It is then followed by our underlying system model and the proposed adaptive solution.

A. Overview of the Existing Technique

Our opportunistic data forwarding protocol [1] works in three phases: (i) receiving and storing data for a certain fixed hold time, (ii) developing specific metrics to evaluate forwarders' data forwarding capability, and (iii) forwarding a train of data packets to the selected next hop forwarder. The goal of the first phase is to receive data packets and stores them until a forwarder discovery is initiated. Each node maintains a local buffer which works in a FIFO manner to store and forward data packets. Each node initiates forwarder discovery in the second phase based on its fixed data hold time. In this phase, a set of candidate forwarders are evaluated based on its link transmission reliability and data reachability. Finally, data packets are forwarded to the selected neighbor back-to-back in one burst. Simulation results in ns2 with realistic underwater environment confirmed that our opportunistic data forwarding protocol outperforms some other existing underwater routing protocols (e.g., VBF [18], DBR [8]) in terms of successful packet delivery, routing overhead and energy consumption. However, as mentioned in Section I, the protocol's performance can be further improved if we calculate data hold time at each node based on its local network conditions and data transmission characteristics. In this paper, we develop an adaptive hold time estimation technique that allows nodes to calculate their data hold time dynamically according to some local network parameters and accumulate data packets accordingly rather than holding data for a fixed amount of time (details in Section III. C).

B. System Model and Network Architecture

In this section, we present the network model and assumptions for our discussions. We assume that sensor nodes are deployed uniformly in the 3D area in underwater. As shown in Fig 1., each node is assumed likely to be a data source (referred to as '*ordinary node*'). They can sense data and relay them towards one or more ordinary nodes acted as data collector (referred to as '*GW*') hop by hop through acoustic communication. '*GW*' nodes relay data collected from other ordinary nodes to one or more mobile sensor nodes (termed as '*ferry nodes*'). Mobility incorporated sensor nodes (e.g., AUVs, ships) are referred to as '*ferry*' nodes in our protocol. Ordinary nodes are anchored to the ocean bottom using a cable/wire and move with ocean current, tide and other environmental factors. After collecting data from '*GW*' nodes, ferry nodes periodically send those data to the onshore base station for further processing. The network can be viewed by a graph $G(V, E)$, where, $V = \{v_1, v_2, ..., v_N\}$ is a set of nodes in a finite dimension of 3D volume, with $N = |V|$ and E is a

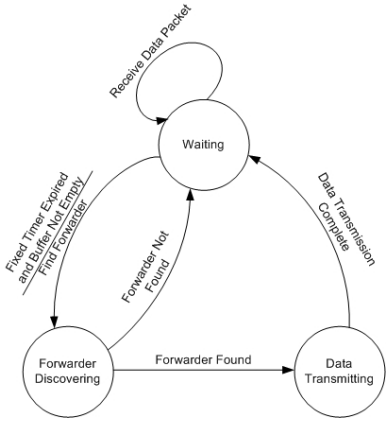

Fig. 2. Fixed Timer for Data Transmission

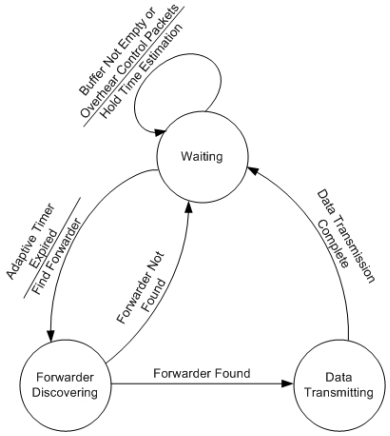

Fig. 3. Adaptive Timer for Data Transmission

set of links among nodes, i. e., e_{ij} equals to 1 if node v_i has a path towards *GWs* through v_j. $S \subset V$ is a set of data sources (ordinary nodes) that sense ocean data and send it to the GW nodes.

C. Adaptive Hold Time Estimation

As mentioned before, sensor nodes maintain a local buffer in our opportunistic data forwarding protocol. Each node i can be any one of the following states: i) Waiting, ii) Forwarder Discovering, and iii) Data Transmitting during its lifetime. Fig. 2 shows the life cycle of a node during its protocol operation using fixed hold time. Initially each node is in the 'Waiting' state. If a packet arrives from an upper layer or from the network, the protocol starts accumulating data packets until a fixed timer running in this node is expired. Once the fixed timer expires and there are a number of packets in its local buffer, the node moves to 'Forwarder Discovering' state. Node *i* then sends a '$Request$' control packet to its one hop neighbors to find a suitable next hop forwarder towards the GW. If no candidate forwarder is found, the node enters into the 'Waiting' state again. Otherwise, a node can receive multiple '$Reply$' packets for a particular request packet in the protocol. After receiving '$Reply$' from multiple candidate forwarders, node i calculates cost for each candidate forwarder j as [1],

$$cost(ij) = \frac{H_j + 1}{LQ_{ij}} \tag{1}$$

Here, LQ_{ij} is the link quality of node j and its uplink forwarding path and H_j is the reachability to the GW (i.e., minimum hop count) using j as next hop forwarder. LQ_{ij} is calculated in the range of [0, 1]. Among multiple candidate forwarders, node i selects the node with minimum cost as the next hop forwarder. Details of the forwarder selection technique can be found in [1]. Node i then switches to 'Data Transmitting' state and starts sending data stored in its local buffer to the selected forwarder. For any node, if it does not receive any '$Reply$' from candidate forwarders, it will re-initiate forwarder discovery process by sending another '$Request$' packet until the maximum number of transmission attempts is exceeded.

Fig. 3 shows the life cycle of a node during its protocol operation using adaptive hold time. In this case, after receiving data packets, each node stores data packets in their local buffer. Each node starts calculating/updating hold time when any of the following events occurs while the node is in 'Waiting' state:

i. Every time one or more data packets enter into the local buffer (i.e., buffer not empty),
ii. One or more control packets are overheard from neighbors,
iii. Forwarder not found and buffer not empty.

Once set, the hold timer will be updated if the node receives more data packets or overhears control packets from its neighbors before the current adaptive hold time expires. Once the timer expires, the node moves to 'Forwarder Discovering' state. If a reliable forwarder is not found within a certain time limit, the node enters into the 'Waiting' state again and updates hold time based on its stored data packets. After forwarder discovery, a node moves to 'Data Transmitting' state and forwards accumulated data to the selected forwarder. Each node takes following factors into considerations before calculating the adaptive hold expiration time (t_e).

1) Delay Experienced by Stored Data Packets (w_t): The delay experienced by a packet at each node adds to the overall end-to-end delay. Thus, the longer a packet spends in a node's buffer, the shorter it should be further held before forwarding. This can be done by limiting the maximum allowable hold time at a node according to average delay experienced by the already stored data packets in buffer. Therefore, this gives an estimate as to when the packet should be forwarded to the next hop towards GW. Each packet entered into the buffer is time stamped. They are used to calculate the average delay, w_t, experienced by the packets in the buffer. Suppose there are n items stored in the local buffer and t_k is the arrival time of k^{th} packet. The average delay, w_t, experienced at current time t, at each node can be calculated as,

$$w_t = \frac{1}{n}\sum_{k=1}^{n}(t - t_k) \tag{2}$$

2) Service Time of Stored Packet (S_t): The longer the line of packets waiting to be serviced, the longer is the delay those packets experience to reach the GW. Therefore, the service time of stored data packets gives an idea about the maximum allowable data packet accumulation time at each node. As the service time increases, the hold time should be decreased to counteract the aforementioned delay. If b_l is the number of data packets currently in a node's buffer that has to be served and μ is the service rate, the service time required for stored packets can be calculated as,

$$s_t = \frac{b_l}{\mu} \tag{3}$$

Algorithm 1 Adaptive Hold Time Calculation

Require: $b_l, \lambda, \mu_e, last\_time, t_e$
Ensure: Updated Hold Time
$h_c \leftarrow 0$
$\delta \leftarrow 0$
if Node receives data packet **then**
  Calculate average waiting delay, w_t using Equation (2)
  Calculate service time of stored packets, s_t using Equation (3)
  Calculate maximum allowable hold time, h_{all} at a node using Equation (4)
  Calculate buffer fill-up time, b_f using Equation (5)
  Calculate current hold duration, h_c using Equation (6)
  if $b_l > 1$ **then**
    $time\_diff = h_c - (current\_time - last\_time)$
    if $time\_diff > 0$ **then**
      if $t_e > (current\_time + time\_diff)$ **then**
        $h_c = time\_diff$
      end if
    end if
  end if
else if Node overhears $Request/Reply$ packet from neighbors **then**
  Calculate channel business from $Quiet$ time, δ using Equation (7)
  if $(current\_time + \delta) < t_e$ **then**
    $\delta \leftarrow 0$
  end if
else if Node fails to discover a forwarder **then**
  repeat step 4 to 8
end if
if $(h_c \neq 0)$ **or** $(\delta \neq 0)$ **then**
  $t_e = current\_time + h_c + \delta$
  $last\_time = current\_time$
end if

Therefore, using equations (2-3), the maximum allowable hold time, h_{all} can be calculated as,

$$h_{all} = \max\{h_{max} - (w_t + s_t), 0\} \tag{4}$$

Here, h_{max} is a system parameter (i.e., the maximum hold time of a packet at any node). It's value can be set by the network administrator according to application requirements.

3) Buffer Fill-up Time (b_f): The time required to fill-up the remaining part of the buffer with incoming packets is an indication about the time a node should wait before forwarding its stored packets. The higher the buffer fill-up time (i.e., more empty spaces in the buffer), the lower the chance of packet drop due to shortage of buffer space. At each node, if packets are entered into the buffer at a rate of λ, b_l is the number of packets currently in the buffer and b_{max} is the maximum buffer size, the buffer fill-up time can be calculated as,

$$b_f = \frac{(b_{max} - b_l)}{\lambda} \tag{5}$$

So, combining (2-5), the current hold duration, h_c, can be calculated as,

$$h_c = \frac{(\mu_e - \lambda)}{\mu_e + 1} \times \left(1 - e^{\frac{-b_f}{h_{max}+1}}\right) \times \frac{2 \times h_{all}}{(hop\_count + 1)} \tag{6}$$

Here, μ_e is the effective service rate and calculated as, $\mu_e = \mu \times LQ_{ij}$, hop_count is the minimum number of hops required to reach the GW. The first factor in equation 6 ensures that the more the arrival rate, the less the data packets wait in the buffer. It reduces the chance of packet drop. At each node, the higher the link quality with the neighbors, the lower is the possibility of packet loss due to link disruption; therefore, packets may be accumulated for a longer time which indicates a larger hold time. Hence, the difference between the effective service rate and the arrival rate of the packets contributes to the data accumulation time (hold time) of the packets in the buffer. The second term factors is the current status of the buffer. The longer a node needs filling-up the buffer, the longer it will be able to hold data packets. It reduces the chance of packet drop due to shortage of buffer space. For example, b_f = 0 indicates a full buffer, which in turn means packets should be forwarded immediately without holding (i.e., h_c = 0). As the value of b_f increases, h_c increases according to the middle part of the equation 6. Finally, the third term in equation 6 ensures that the further a node is from the GW, the less amount of time it should hold data packets before forwarding. This ensures that nodes one hop away from the GWs are allowed to hold data packets for longest time and this allowable hold time decreases for nodes farther away from the GW. The reason is that the end-to-end delay increases with the number of hops and this factor attempts to compensate that to certain extent by adjusting hold time according to a node's distance to GW. Finally, the adaptive expiration time (t_e) of h_c can easily be calculated by adding it to the current time.

The expiration time (t_e) can be further adjusted according to the channel business, which estimates how busy a channel is/will be. This information is required in the calculation of t_e to reduce the chance of collision. Channel business can be estimated from the information in the forwarder discovery packets of neighboring nodes. The forwarder discovery process ($Request - Reply$ exchange) is designed to determine a reliable next hop among a set of candidate next hop forwarders. When sending a '$Request$' packet, each node includes a '$Quiet$' time (δ), in the packet. Any node receiving '$Request$' packet checks if it has valid neighbor information to reach the gateway. Candidate forwarders then send a '$Reply$' packet to let the sender node know about their ability to act as a forwarder node. Candidate forwarders also include a '$Quiet$' time (δ), in the '$Reply$' packet. The '$Quiet$' time (δ) can be calculated as,

$$\delta = \begin{cases} T_{req} + T_{rep} + T_{data} + 3 \times T_{prop} & \text{if } Request \\ T_{rep} + T_{data} + 2 \times T_{prop} & \text{if } Reply \end{cases} \tag{7}$$

Here, T_{req} is the amount of time a sender node requires to transmit the '$Request$' packet, T_{rep} is the time required by a candidate forwarder to transmit '$Reply$' packet, T_{data} is the amount of time required by a node to send its stored data packets, and T_{prop} is the maximum one hop propagation delay of a node. In the '$Request$' packet, 3 $\times$ maximum propagation delay (T_{prop}) consists of the propagation delay to send '$Request$', '$Reply$' and '$Data$' packets. After overhearing '$Request$' packet, each node knows that it has to refrain from any sort of transmission and keep holding the data for a further period of δ to reduce the chance of collision with neighboring nodes. Therefore, each node overhearing the '$Request$' packet adjusts its adaptive hold expiration time, t_e, based on δ.

TABLE I. SIMULATION SETTINGS

Propagation Speed	1500 m/s
Transmit Power	2 Watts
Antenna Model	Omni-directional
Maximum Transmission Range	120 meters
Packet Size	200 Bytes
Initial Energy	1000 J
Traffic Model	CBR

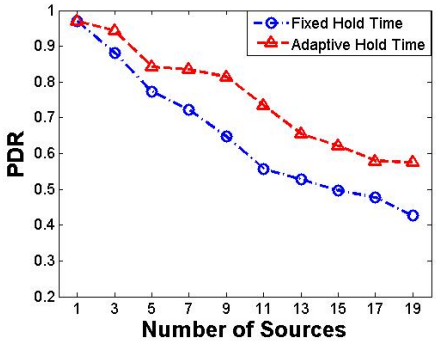

Fig. 4. PDR vs Number of Sources

When a sender node i receives '$Reply$' packets from multiple forwarders, it evaluates the best next hop forwarder *'on-the-fly'* based on a selection metric of equation 1 and then starts transmitting data packets to the selected forwarder. Any node l, located within the transmission range of the candidate forwarder, overhears the '$Reply$' packet and knows that there might be a chance of collision at the candidate forwarder's end if it transmits anything during the period of δ specified in the '$Reply$' packet. Thus, node l adjusts its hold expiration time, t_e according to δ. In case of '$Reply$' packet, 2×maximum propagation delay (T_{prop}) includes the propagation delay to send '$Reply$' and '$Data$' packets.

The calculation of adaptive hold expiration time (t_e) at each node is described using Algorithm 1. Before presenting the formal algorithm, we introduce the notations used: b_l denotes the number of data packets currently in a node's buffer, λ is the packet arrival rate, and μ_e is the effective service rate, $last\_time$ is the latest time when a node sets its hold time and t_e is the expire time of current hold time. Line 1-2 initializes the variables in the algorithm. Line 3-16 calculates current hold duration, h_c according to the received data packets and updates it based on previous hold expiration time, t_e. Line 17-21 calculates δ according to overheard control packets and updates it based on previous hold expiration time, t_e. Line 22-24 updates h_c if a node fails to discover a forwarder. Finally, line 25-28 calculates hold expiration time, t_e for each node based on calculated h_c and δ. During this time, each node is allowed to accumulate data packets in its local buffer and prevents itself from transmission to reduce collision.

IV. PERFORMANCE EVALUATION

To verify the effectiveness of the proposed adaptive technique, we evaluate the performance of both fixed and adaptive versions through simulations in ns2.

A. Simulation Settings

The ns2 platform fully considers the underwater acoustic environment by modeling the channel in the physical layer as in [13]. Carrier Sense Multiple Access (CSMA) protocol is used as an underlying MAC protocol. Each simulation runs for 900 seconds. For all topologies, the results were averaged from 10 simulation runs. Simulation settings are shown in Table 1. We used the following metrics to evaluate the performance of our opportunistic forwarding protocol for both the fixed and adaptive cases. The packet delivery ratio (PDR) is defined as the ratio of the number of packets successfully received at the GW nodes to the number of packets transmitted from the source nodes. The end-to-end delay is the average time taken by a packet to reach from a source to any of the GWs. Energy consumption per successful data packet reception is evaluated by dividing the total energy consumed by the sensor nodes during the forwarding of data packets from source to GW nodes by the number of packets successfully received at the GW nodes. Routing overhead is the amount of control traffic expended as a fraction of total traffic.

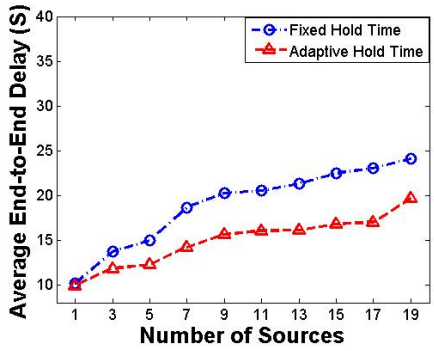

Fig. 5. End-to-End Delay vs Number of Sources

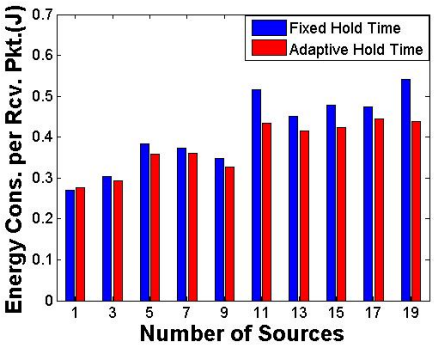

Fig. 6. Energy per Received Packet vs Number of Sources

B. Simulation Results and Analysis

1) Source Density: Many underwater applications such as environment monitoring [2], [5] require sensor nodes to

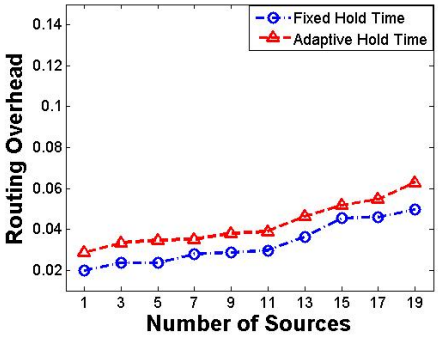

Fig. 7. Routing Overhead vs Number of Sources

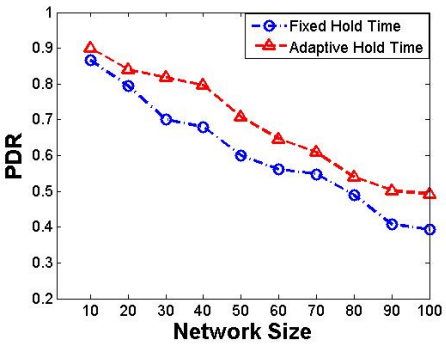

Fig. 8. PDR vs Network Size

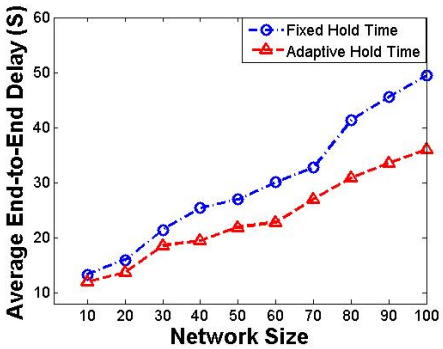

Fig. 9. End-to-End Delay vs Network Size

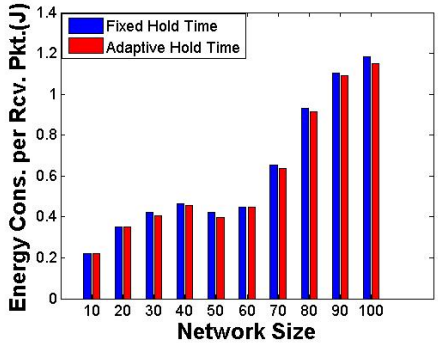

Fig. 10. Energy per Received Packet vs Network Size

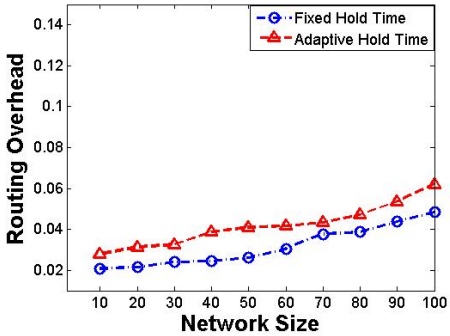

Fig. 11. Routing Overhead vs Network Size

sense data at different points in the network and report those data to the GW nodes for further analysis (e.g., average temperature/salinity of a region). The GWs can perform this analysis more accurately if it can capture related data from a variety of sources within the region of interest. In order to asses the protocol's performance with various sources, in this simulation, 20 sensor nodes (19 ordinary nodes and one GW) were deployed uniformly within 225×225×225 m^3 area underwater. Simulations were performed by varying the number of source nodes from 1 to 19 nodes. The simulation result for PDR under various source densities is shown in Fig. 4. In each scenario, source nodes generate one packet per second. As shown in Fig. 4, both the fixed and adaptive versions of our protocol show a decreasing trend in PDR as the number of active source nodes increases. At high source densities, the number of forwarding nodes increases which in turn causes large number of collisions in the network and, therefore, decreases PDR. However, the decrement is considerably lower in our protocol with adaptive hold time technique compared to the fixed case. At relatively high source density (e.g., 11 sources), our protocol with adaptive hold time technique achieves higher PDR (e.g., 0.7352) than the protocol with fixed hold time (e.g., 0.5575). The reason is that, with adaptive approach, the protocol takes own as well as neighbors' data transmission into consideration during the calculation of hold time at each node. This reduces the chance of collision in the network and thus, improves PDR. Therefore, Fig. 4 shows that our protocol with adaptive hold time is more effective than the previous version with fixed hold time.

Fig. 5 presents how the source density impacts on the end-to-end delay and indicates that the end-to-end delay increases in both fixed and adaptive cases as the source density increases. This is due to the increased network traffic load and traffic flow from different points in the network. However, the end-to-end delay in the adaptive case is significantly lower compared to the fixed case. For the latter case, each sensor node holds data for a fixed amount of time (10s for all nodes in this simulation) irrespective of the neighborhood network conditions. In contrast, for the former, each sensor node calculates its hold time dynamically according to its necessity based on data packet arrival/effective service rate, delay experienced by each packet at node's buffer, neighbors' transmission, reachability information. This in turn reduces per hop waiting delay which contributes to reduce the overall end-to-end delay. Therefore, the calculated hold time at each node in adaptive case is always less than the hold time at fixed case.

The energy consumption per successful data packet received at the GW node is investigated in Fig 6. It is seen from the figure that the total energy spent per successful packet reception is higher in fixed hold time case compared to the adaptive case. This is due to efficient calculation of hold time in the latter case considering neighbors' transmissions which reduces collisions. The reduced collision increases the chance of successful data delivery and consumes less energy for per successful data packet delivery. Fig. 7 exhibits the effect on routing overhead of our protocol (in both cases). The routing overhead increases for both cases with the number of sources. Increased number of sources increases the number of control packets to find more routing paths in the network. However, compared to the scheme with fixed hold time, the routing overhead increases in our protocol with adaptive hold time. This is because the calculated adaptive hold time at each node decreases as the nodes get farther from the GW node. This causes far away nodes to initiate neighbor discovery more frequently compared to the fixed case and hence the overhead rises. However, as the adaptive technique performs much better in all three other metrics, this moderate increase in overhead is justified.

2) Scalability: Many underwater applications require deploying a large number of sensor nodes over a certain geographic area of interest. To support those applications, the network size varies from 10 to 100 nodes in this simulation. For each network size, 20% nodes are chosen randomly to act as data sources. Thus, the network density remains the same for different network sizes, i.e., the geographic area ($150\times150\times150$ m^3 - $600\times600\times600$ m^3) of the network was increased proportionally according to the number of nodes. Similar to source density, data packets are generated at an interval of 1s. As shown in Fig. 8, the PDR decreases as the network size increases in our protocol for both fixed and adaptive case. This is due to the same reason already explained in Fig. 4. However, PDR is considerably higher in adaptive case than that of the fixed case. This is due to the same reason already explained in relation to Fig. 4. Fig. 9 shows the impact of end-to-end delay on our protocol. The end-to-end delay increases in both cases as the network size increases. However, the end-to-end delay is significantly lower and the rate of delay increment with network size is less steep in the adaptive case compared to the fixed case. The superior performance in the former case is due to its effective calculation of hold time dynamically as described in Fig 5. The less accumulation time at each node in the adaptive case compared to the fixed case reduces the end-to-end delay in the protocol. Fig. 10 shows the effect of network size on the energy consumption per successful packet reception at the GW. As the network size increases, more and more traffic are generated in the network, which results in greater dissipation of energy due to the increased traffic load along with the increased overhead as the network becomes bigger in both cases. However, it is observed from Fig. 10 that energy consumption per successful reception at the GW nodes is almost the same (e.g., Network size of 60) or lower (e.g., Network size of 80) in the adaptive case compared to the fixed case. The energy efficiency in the former case is achieved due to the same factors explained in Fig. 6. The effect of routing overhead on the performance of the protocol for both cases is observed in Fig. 11. The overhead increases in both cases as the network size increases. The reason is that, the collision domain increases with the network size, i.e., data packets need to travel through more hops to reach the GW which increases the chance of collision. However, the increase in overhead for adaptive case is slightly higher compared to the fixed case due to the similar phenomenon observed in Fig. 7. However, as alluded before, the benefit achieved in terms of other metrics weigh more than the impact of increasing overhead.

V. Conclusion

In this paper, an adaptive approach has been proposed to opportunistic data forwarding protocol in UASNs. The proposed technique has alleviated the end-to-end delay problem due to fixed hold time approach in previously proposed data forwarding protocol [1]. Existing technique with the fixed hold time is designed for delay tolerant applications and achieves high packet delivery ratio with less overhead and energy consumption. However, it is observed that each node in the network does not need to hold data for the same amount of fixed time. It increases the end-to-end delay and also increases the chance of collisions in the network. Therefore, an adaptive technique has been proposed which reduces end-to-end delay by calculating hold time at each node considering the local network conditions rather than using a fixed hold time. Our adaptive solution has also taken neighbors' data transmissions into consideration to further improve data delivery. Simulation results show that compared to fixed hold time, the adaptive technique achieves considerably higher PDR with less energy consumption per successful packet delivery and improved end-to-end delay. Determining data retrieval strategy for the ferry nodes to further maximize data delivery remains the focus of our future study.

References

[1] N. Nowsheen, G. Karmakar, and J. Kamruzzaman, "An Opportunistic Message Forwarding Protocol for Underwater Acoustic Sensor Networks," Proc. The 19th Asia Pacific Conference on Communications (APCC), pp. 208–213, 2013.

[2] 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.

[3] I. F. Akyildiz, D. Pompili, and T. Melodia, "State of the Art in Protocol Research for Underwater Acoustic Sensor Networks," ACM SIGMOBILE Mobile Computing and Comm. Rev. , vol. 11, pp. 11–12, 2007.

[4] M. Chitre, S. Shahabudeen, and M. Stojanovic, "Underwater Acoustic Communication and Networks: Recent Advances and Future Challenges," Marine Technology Society Journal, vol. 42, no. 1, pp. 103–116, 2008.

[5] D. Pompili and I. Akyildiz, "Overview of Networking Protocols for Underwater Wireless Communications," IEEE Communications Magazine, vol. 47, no. 1, pp. 97–102, 2009.

[6] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internet," Proc. ACM SIGCOMM, pp. 36–34, 2003.

[7] J. Partan, J. Kurose, and B. N. Levine, "A Survey of Practical Issues in Underwater Networks," Proc. First ACM International Workshop on Underwater Networks (WUWNet), pp. 17–24, 2006.

[8] H. Yan, Z. J. Shi, and J.-H. Cui, "DBR: Depth-based Routing for Underwater Sensor Networks," Proc. 7th International IFIP Networking, pp. 72–86, 2008.

[9] A. Wahid and D. Kim, "An Energy Efficient Localization-free Routing Protocol for Underwater Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 2012, id. 307246, pp. 1–11, 2012.

[10] R. Javidan and H. Rafiee, "A New Energy Efficient and Depth based Routing Protocol for Underwater Sensor Networks," British Journal of Science, vol. 8, no. 1, 2013.

[11] Z. Li, N. Yao, and O. Gao, "Relative Distance-based Forwarding Protocol for Underwater Wireless Networks," International Journal of Distributed Sensor Networks, vol. 2014, no. 173089, pp. 1–11, 2013.

[12] A. Wahid, S. Lee, and D. Kim, "An Energy-Efficient Routing Protocol for UWSNs using Physical Distance and Residual Energy," Proc. IEEE OCEANS, pp. 1–6, 2011.

[13] A. F. Harris and M. Zorzi, "Modeling the Underwater Acoustic Channel in ns2," Proc. International Conference on Performance Evaluation Methodologies and Tools - NSTools, 2007.

[14] A. Vahdat and D. Becker, "Epidemic Routing for Partially Connected Ad hoc Networks," Technical Report, Dept. of Comp. Sc., DUKE University, Durham, NC, 2000.

[15] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," Proc. ACM SIGCOMM, pp. 252–259, 2005.

[16] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic Routing in Intermittently Connected Networks," Mobile Computing and Communications Review, vol. 7, no. 3, pp. 19–20, 2003.

[17] J. Legyay, T. Friedman, and V. Conan, "DTN Routing in a Mobility Pattern Space," Proc. ACM SIGGCOMM, pp. 276–283, 2005.

[18] P. Xie, J.-H. Cui, and L. Lao, "VBF: Vector-based forwarding protocol for underwater sensor networks," Proc. IFIP Networking, pp. 1216–1221, 2006.