Received February 15, 2021, accepted March 19, 2021, date of publication March 26, 2021, date of current version April 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3069077

Coding-Aware Opportunistic Routing for Sparse Underwater Wireless Sensor Networks

DANFENG ZHAO, GUIYANG LUN, AND RUI XUE
College of Information and Communication Engineering, Harbin Engineering University, Harbin 150001, China

Corresponding author: Guiyang Lun (lunguiyang@hrbeu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61371099.

ABSTRACT In underwater wireless sensor networks (UWSNs), the sensor nodes are sparsely deployed over a large sea area due to their high design cost and high manufacturing cost. Opportunistic routing protocols are a promising forwarding technique for various UWSNs. However, many opportunistic routing protocols suffer from the void problem in sparse underwater scenarios. Hop count-based opportunistic routing protocols inherently alleviate this problem by periodically maintaining the topological information of sensor node. Nevertheless, the robustness of these protocols degrades due to channel variation and node movement in UWSNs. In this paper, we propose a coding-aware opportunistic routing method for sparse UWSNs (CORS). In CORS, we use topological information to adaptively expand the candidate set. On this basis, a forwarding with opportunistic coding strategy is developed to join interflow network coding and opportunistic forwarding in CORS. In addition, we design a sliding window-based coding algorithm to provide effective coding gains with low coding overhead. Then, a sliding window-based decoding algorithm is designed to reduce decoding overhead. Simulation results show that CORS significantly improves upon the network performances of existing protocols in various scenarios.

INDEX TERMS Underwater wireless sensor networks, opportunistic routing, hop count, network coding.

I. INTRODUCTION

Underwater wireless sensor networks (UWSNs) have attracted much attention in scientific research and industrial applications [1], [2]. UWSNs are composed of sensor nodes deployed in regions of submarine environments to monitor climates, trace pollution, assist navigation, prevent disasters, etc. In many scenarios, aqueous systems are dynamic, and events occur within the water mass since it advects and disperses within special regions. A typical solution is to use a swarm of mobile sensors to construct observation systems [3]–[6]. Generally, mobile nodes are sparsely deployed over a large area due to their high design cost and high manufacturing cost [7], [8]. The information of each sensor node is transmitted to the sink node through multihop transmission and then forwarded to the onshore center via RF transmission.

The function of the routing protocol in a UWSN is to establish a forward path toward the sink node. Traditional routing protocols are not efficient and not feasible in some scenarios due to the complexity of the channel, high latency and variable topology in harsh underwater environments [2]. In contrast, opportunistic routing (OR) is an effective solution for these problems, as it can establish a path to the sink by taking the broadcast nature of wireless media and opportunistic forwarding into consideration [9], [10].

The void problem is the main factor that degrades the network performance in a given sparse UWSN [11]. Among various OR protocols, hop count-based OR routing protocols can effectively alleviate the void problem due to the application of the topological information that is periodically established by the sink [12], [13]. However, in sparse UWSNs, the candidate set is usually small, and the void problem occurs heavily due to node movement and channel variation. Node movement can disable the link between a transmission pair. As shown in Fig. 1, node D fails to forward packet $d2$ to node F because link DF fails with the movement of node F. Since node D only has one neighbor with a smaller hop count (HC) than that of itself, packet $d2$ cannot reach sink1. Moreover, channel variation influences the reliability of topology establishment and maintenance. As shown in Fig. 1, the hop count of node G is smaller than that of node E. The channel state of link EG is best in the topology establishment stage. However, as the

The associate editor coordinating the review of this manuscript and approving it for publication was Emrecan Demirors.

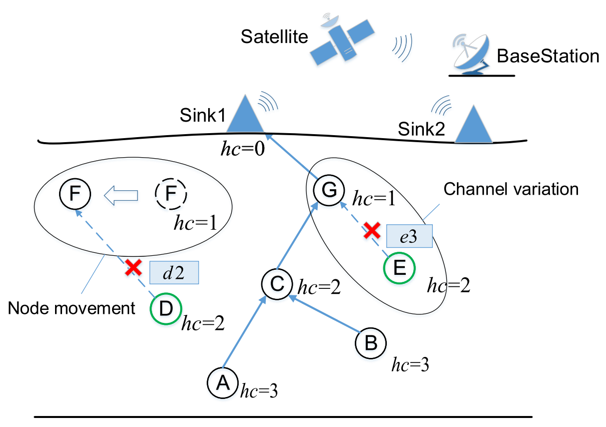


FIGURE 1. The void problem due to node movement and channel variation in sparse UWSNs.

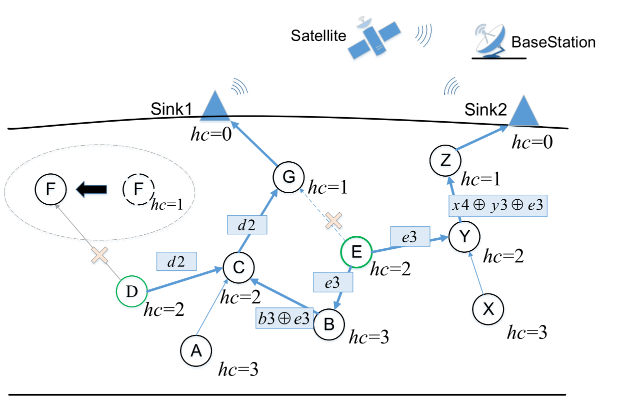


FIGURE 2. Candidate set expansion and opportunistic coding strategy.

channel worsens, node G cannot properly receive packet $e3$ from node E. Thus, packet $e3$ cannot reach Sink1 since node E has only one neighborhood with a smaller hop count than that of itself. These factors heavily influence network performance in scenarios with few nodes in the candidate set.

Candidate set expansion is a effective method to improve the delivery performance of a sparse UWSN. Unlike typical hop count-based OR protocol, we incorporate the nodes with equal or high hop counts in the candidate set. As shown in Fig. 2, node C has the same hop count as node D. By using the opportunistic forwarding algorithm, node C can forward the packet overheard from node D. Then, packet $d2$ can reach Sink1 across node G. However, the neighboring nodes with equal or higher hop counts directly forward the packet, which would introduce unnecessary transmissions. Thus, an opportunistic forwarding strategy is required to adaptive to dynamic network topology.

Network coding is a natural method for forwarding one packet integrated by many packets. [14]. By using network coding, intermediate nodes can encode packets from different traffic flows to reduce redundant transmission [14], [15]. In addition, network coding can improve transmission reliability in networks with poor channel conditions by integrating diverse packets from different nodes into the redundant transmission rather than merely forwarding the duplicate packets [16], [17]. To recover original packets, a node requires the coded packets whose quantity is fewer than that of duplicate forwarding. This is robust in the networks with dynamic network topology. As shown in Fig. 2, node B encodes incoming packet $e3$ with local packet $b3$ into coded packet $b3 \oplus e3$. Similarly, node Y generates packet $x4 \oplus y3 \oplus e3$ by encoding packet $e3$ with local packet $x4 \oplus y3$. Thus, packet $e3$ can be decoded if the base station receives one of packets $b3$ and $x4 \oplus y3$. In contrast, the base station can successfully decode packet $b3$ after decoding packet $e3$ since packet $b3$ fails to reach it beforehand. However, if node B directly forwards $e3$ without coding with packet $b3$, packet $b3$ still cannot be collected by the base station. Thus, the delivery performance in a sparse UWSN can be improved by designing a reasonable opportunistic coding mechanism based on interflow network coding in the network layer.

Several protocols have been designed for multihop wireless mesh networks by integrating network coding with opportunistic routing [17]–[20]. However, these protocols aim at improving network performance for networks with multiple cross-traffic flows or with multicast traffic, and they cannot effectively optimize the routing problem in the networks with unidirectional flows to the sink nodes. Thus, we develop a Forwarding with Opportunistic Coding (FOC) strategy based on hop count-based candidate set expansion method and interflow network coding.

In FOC strategy, each intermediate node calculates the probability of opportunistic coding with the topological information conveyed in the incoming packet. Each node encodes multiple packets from different source nodes. In addition, we use the source address and sequence identifier (SID) to distinguish each encoding symbol since that multiple packets generated from the same source node can reach the immediately adjacent node within a small time interval. However, the number of encoding symbols becomes larger as each source node continuously generates packets. The overgrowth of encoding symbols not only increases coding overhead but also introduces additional decoding delay. An method to resolve this problem is only coding the packets from the associated region since this can significantly limit the amount of input symbols. However, it is not feasible to divide the coding region by spatial location in mobile scenarios.

To provide effective coding gains with low coding overhead and low decoding overhead, we develop a novel Sliding Window-based Coding (SWC) algorithm. Unlike conventional sliding window-based coding, SWC is a coding algorithm designed for interflow network coding that is used in the networks with unidirectional flows. In SWC algorithm, a node dynamically adjusts its coding region according to the SIDs of local buffered packets and the incoming packet. Moreover, a maximum buffering time interval is used to restrict the input packets. In addition, we use a maximum coding degree to control the sparsity of the coding vector. We set a constant value of the coding vector size. Thus, the decoding matrix is constant and cannot be affected by the generation of new data packets in the sensor nodes. Furthermore, we design a Sliding Window-based Decoding (SWD) algorithm to reduce the decoding overhead introduced by conventional decoding process. At last, we integrate the SWC algorithm into the FOC strategy. On this basis, we propose a coding-aware opportunity routing method for sparse UWSNs (CORS).

The rest of the paper is organized as follows. In Section 2, the related work with respect to typical OR protocols in UWSNs is summarized. Section 3 defines the network architecture and channel model exploited in this paper. In Section 4, we illustrate the key components of CORS, namely, network initialization and maintenance, forwarding with an opportunistic coding strategy, a sliding window-based coding algorithm, a sliding window-based decoding algorithm, and a backoff strategy. Then, we analyze the proposed protocols in Section 5. In Section 6, we present a performance evaluation and discussion. Finally, we conclude the paper in the last section.

II. RELATED WORKS

In OR protocols, candidate selection is generally based on geographic information or network topology [9]. Thus, we categorize recent OR protocols into geographic-based, pressure-based and hop count-based OR protocols.

A. GEOGRAPHIC-BASED ROUTING PROTOCOLS

Vector-based forwarding (VBF) was the first opportunistic routing protocol developed for UWSNs based on coordinate information [21]. In VBF, the candidate set is divided by a pipeline from each sensor node to a sink node. Immediate nodes receive a packet judging whether they cooperate with forwarding this packet by using the coordinates of the source, the sink and the current node. In VBF, only the node in the pipeline can forward a packet. On the basis of VBF, hop-by-hop vector-based forwarding (HH-VBF) optimizes the candidate set by adjusting the pipeline toward the sink node [22]. Adaptive hop-by-hop vector-based forwarding (AHH-VBF) resolves the redundant transmission problem in an underwater sparse network by dynamically adjusting each forward region [8]. Geographic and opportunistic routing with depth adjustment-based topology control for communication recovery over void regions (GEDAR) uses the 3D location of each node for selecting a candidate set and for estimating link reliability to select the optimal forwarder [10]. In addition, GEDAR solves the void problem by adjusting the depth positions of corresponding nodes. The game-theoretic routing protocol optimizes the candidate set with location information and designs a game-theory-based forwarding strategy (GTRP) [23]. These protocols based on three-dimensional (3D) locations can address the hidden terminal problem and alleviate the void problem to some extent. However, the reliability of the applied routing algorithm heavily depends on the accuracy of acoustic positioning since the 3D location of each node is obtained through acoustic positioning. Moreover, the use of acoustic positioning adds additional energy overhead and introduces additional interference. Therefore, these protocols are only suitable for network scenarios with slight link variations and regional node movements.

B. PRESSURE-BASED ROUTING PROTOCOLS

Since many underwater sensors are integrated with pressure sensors, each node can obtain depth information without introducing communication overhead due to expensive distributed localization. In view of this, a series of opportunistic underwater routing protocols have been developed based on depth information. Depth-based routing (DBR) is a typical receiver-side-based opportunistic routing protocol that uses depth information to select candidate sets [24]. A depth threshold is used to select the candidate set. In DBR, the node with the lowest depth has the highest priority to forward packets. On the basis of DBR, the energy-efficient depth-based routing (EEDBR) and weighting depth and forwarding area division DBR (WDFAD-DBR) protocols were developed [25], [26]. EEDBR considers residual energy during the process of candidate set selection, and this optimizes the network lifetime while reducing the packet delivery ratio. However, DBR and EEDBR suffer from the void problem, as only 1-hop information is considered. WDFAD-DBR solves the local optimization problem and the void problem via depth weighting and forwarding region division. However, the algorithm for forwarding region division relies on the accuracy of acoustic positioning for each node. Moreover, as the network size increases, a large number of control messages interfere with information transmission. Thus, WDFAD-DBR is more suitable for networks with lower traffic. Hydraulic pressure-based anycast routing (Hydrocast) divides potential forwarding nodes into multiple candidate sets [27]. In addition, the distance information is used to estimate the channel between each pair of neighbors to calculate the normalized advance (NADV) of each candidate set. However, the reliability of the NADV relies on the measurement accuracy of the relative position of each node. Moreover, the theoretical model used to estimate the packet loss ratios of channels cannot be used directly in actual underwater network scenarios. The protocols in this category can effectively address the routing problems in dense networks, since they use the depth information together with other information of neighboring nodes. However, they require more transmission redundancy to address the void problem and to improve the delivery performance in networks with time-varying channel condition.

C. HOP COUNT-BASED ROUTING PROTOCOLS

Hop count-based routing protocols for UWSNs are commonly based on the hop count information established by broadcasts from sink nodes. The hop-by-hop dynamic addressing-based (H2-DAB) protocol is an opportunistic routing approach with a multiple-sink architecture [28]. Each node records the distance gradients to two adjacent sink nodes during every updating interval. In H2-DAB, the neighbor node with the smallest hop count is placed into the candidate set. Each node selects the next-hop forwarder by negotiating with the node in the corresponding candidate set. However, H2-DAB fails to use the broadcast nature of wireless media during forwarding. Moreover, the forwarding process is inefficient since H2-DAB exploits sender-side coordination. Two-hop acknowledgment (2H-ACK) routing introduces a two-hop feedback mechanism on the basis of H2-DAB [29]. In 2H-ACK routing, each forwarder sends feedback to inform the last forwarder node after receiving the request reply from the node with the smallest hop count. Although the transmission reliability is optimized, 2H-ACK routing fails to use the broadcast nature of wireless media as well. Moreover, the loss of feedback information increases the amount of unnecessary redundant transmission in the network. Void-aware pressure routing (VAPR) chooses the next-hop forwarder by using hop count, depth and sequence number information [30]. In VAPR, the division of multiple candidate sets is accomplished by a clustering algorithm based on node distances, which can effectively optimize transmission in a dense network. However, the forwarding algorithm relies on the accurate division of the coordinate set, which relates to the locations of two-hop neighbor nodes and the corresponding link information between each pair. The network performance degrades heavily as nodes frequently move or in sparse scenarios. In addition, topology control may affect data transmission at high traffic rates. Inherently void-avoidance routing (IVAR) uses depth information and hop counts to calculate the candidate set to bypass void areas [12]. In IVAR, the distance between each node is estimated by the received signal strength, but this is ineffective due to the complexity and variability of the underwater channel. Distance vector-based opportunistic routing (DOVR) exploits hop count information for establishing a distance vector to improve the void problem and the detoured forwarding problem [13]. In DOVR, the broadcast nature of the wireless medium is effectively exploited to accomplish opportunistic forwarding since the adjacent nodes with the lowest hop counts are all added to the candidate set. However, the candidate set may be invalid within the update interval due to channel variation and node movement. Although this kind of protocol can avoid the void problem to some extent, topology maintenance may consume additional energy. In addition, the network performances of these protocols may be sensitive to dynamic changes in network topology.

III. SYSTEM MODEL

In this section, we first describe the architecture of the underwater network considered in this paper. Then, we review the underwater channel model and derive the signal-to-interference plus noise ratio (SINR) used in the simulation.

A. NETWORK ARCHITECTURE

In this paper, we consider a 3D mobile underwater network scenario, as shown in Fig. 1. The network consists of two types of nodes: sensor nodes and sink nodes. The sensor nodes are isomorphic nodes that have the ability to collect, send, and forward information. Each sensor node deployed at a different depth can move with the current to collect data in sea areas [5]. The sink nodes, uniformly located on the surface of the network, are integrated with an acoustic modem and a wireless modem to communicate with a satellite. Each node transmits information to the sink node directly or through multihop transmission. Then, the information is forwarded to a satellite and relayed to the base station via RF transmission.

B. CHANNEL MODEL

The attenuation of a signal contains three parts: large-scale path gain, spreading loss, and absorption attenuation [31]. We calculate the total attenuation over a distance d for a signal of frequency f as below.

$$A(f,d) = g_p d^k \alpha(f)^d \tag{1}$$

where g_p represents the large-scale gain with a log-normal distribution $10lg(g_p) \sim N(0,\sigma^2)$. k is the spreading factor that describes the extended geometric shape ($k = 2$ for spherical spreading, $k = 1$ for cylindrical spreading). The absorption coefficient $\alpha(f)$ is expressed empirically using Thorp's formula in dB/km for the carrier frequency f in kHz [32] as:

$$10log\alpha(f) = 0.11\frac{f^2}{1+f^2} + 44\frac{f^2}{4100+f} + 0.000275f^2 + 0.003 \tag{2}$$

We model the ambient noise in an underwater environment by the following empirical formula:

$$N(f) = N_t(f) + N_s(f) + N_w(f) + N_{th}(f) \tag{3}$$

where $N_t(f)$, $N_s(f)$, $N_w(f)$, and $N_{th}(f)$ denote the power spectral densities (p.s.ds) of turbulence, shipping, waves, and thermal noise, respectively. The empirical formulas of the four p.s.ds in dB re μPa per Hz as functions of frequency in kHz are presented as follows:

$$\begin{aligned}
10\lg N_t(f) &= 17 - 30\lg f\\
10\lg N_s(f) &= 40 + 20(s-0.5) + 26\lg f - 60\lg(f+0.03)\\
10\lg N_w(f) &= 50 + 7.5w^{1/2} + 20\lg f - 40\lg(f+0.4)\\
10\lg N_{th}(f) &= -15 + 20\lg f
\end{aligned} \tag{4}$$

where s is the shipping activity factor ranging between 0 and 1, while w is the wind speed in m/s.

Due to the superposition of the interfering packets from other nodes, we divide each packet into M segments [33]. Then, the SINR of each segment in terms of the carrier frequency f can be calculated as below:

$$SINR_m = \int_B \frac{p_s(f)/A(f,d)}{N(f)+I_m(f)}df \tag{5}$$

where $p_s(f)$ is the power density of the transmitter for carrier frequency f. B is the bandwidth in Hz. $I_m(f)$ denotes the superposition of the interference power density of the packets in the mth segment, and this can be evaluated as

$$I_m(f) = \sum_{i=1}^{N_m} \frac{p_i(f)}{A(f, d_i)} \tag{6}$$

where N_m is the number of interference nodes in the mth segment. d_i is the distance between the ith interference node and the current node. $p_i(f)$ is the power density of the ith interference in the mth segment for carrier frequency f. For simplicity, we assume $p_s(f)$ is a constant function and that $p_i(f)$ equals $p_s(f)$.

Finally, we use the minimum value of the SINR among all segments in the packet to represent the SINR of the packet itself $SINR = min(SINR_m)$.

IV. CORS PROTOCOL

In the CORS protocol, the topological information of every sensor node is established and updated through network initialization and maintenance phases. Then, during the forwarding phase, each node executes forwarding with an opportunistic coding strategy to send its own packets or relay incoming packets. In the coding process, a sliding coding algorithm is used to encode the incoming packet with buffered packets. Whenever receiving a packet, the base station tries to decode new information with the buffered packets by using a sliding window-based decoding algorithm. Moreover, to alleviate the interference between the control packet and information packet, CORS exploits a backoff strategy to wait to forward outgoing packets.

We assume that the hop count of the current forward node i is hc_i. This node occupies a candidate set C_i, which contains all possible next-hop nodes. We divide all the nodes in the candidate set C_i into three subsets based on their hop count information: (1) Subset L, where the hop counts of the nodes are smaller than hc_i; (2) Subset S, where the hop counts of the nodes are the same as hc_i; and (3) Subset H, where the hop counts of the nodes are higher than hc_i.

A. NETWORK INITIALIZATION AND MAINTENANCE

Each node initializes and updates its hop count information as follows. After the network finishes deployment, every sink node and sensor node initialize their hop count informations to 0 and ∞, respectively. Then, each sink periodically updates the network topology by broadcasting a beacon packet, which contains the hop count information of the sender and the broadcast identification (BID) of this round. The intermediate node that receives the beacon packet first updates its HC and local BID under if the BID is higher than or equal to the local BID. Then, the intermediate node rebroadcasts a beacon packet that carries its HC and current BID. This procedure continues until the node farthest away from the sink nodes successfully updates its topological information. To reduce the interference induced during topology establishment, each node sets a random jitter for broadcasting a beacon packet [10].

To periodically update the topological information for each node, we introduce a beacon broadcast interval T_B. After broadcasting a beacon packet, the sink sets a waiting timer with a time interval of T_B before triggering the next beacon. The update interval should be small enough to ensure effective connectivity despite node movements and link variations between each transmission pair. However, broadcasting beacon packets frequently can increase the probability of collision and introduce communication overhead. In addition, the broadcast mechanism cannot completely ensure the reliability of the topology update process due to unreliable links. Therefore, we introduce an implicit topology refresh mechanism to maintain the topological information. In this mechanism, each node refreshes its topological information by using the data packet overheard from the nodes in subset L.

CORS uses a valid period of the topological information T_{valid} to eliminate invalid topological information in a timely manner. A node that fails to update its HC during T_{valid} is degraded to a void node with a hop count of ∞. Meanwhile, it empties its neighbor table. This mechanism prevents invalid nodes from participating in forwarding. On this basis, CORS introduces a timeout reconnection mechanism to connect to the network in time. A void node that fails to update its topological information during T_{valid} broadcasts a connection request (CR) packet, which includes the node address and local BID. Then, the surrounding nodes with valid HCs that receive this CR packet return a connection acknowledgement (CA) packet to help the void node to reconnect to the network.

B. FORWARDING WITH OPPORTUNISTIC CODING

In CORS, each node occupies two buffers: one is a sending buffer B_s for buffering packets to be sent when the timer expires; the other is a temporary buffer B_t for buffering the packets received from the sparse region. Each buffer can be represented as

$$B_k = \{P_i | i = 1, 2, \ldots, N_k\} \tag{7}$$

where P_i is the ith packet in buffer B_k, while N_k is the number of packets in this buffer.

Once it has received data packet P_{rx}, a node executes Algorithm 1. First, the node checks whether P_{rx} has been recorded. If recorded, it is discarded (line 3). Otherwise, the node checks whether P_{rx} is listed in each buffer (lines 4 and 13). If P_{rx} has been stored in buffer B_t, the node discards the corresponding packet from this buffer on the condition that P_{rx} is overheard from the nodes in subset L (lines 6 and 7). Otherwise, the node waits to forward P_{rx} if it is not in the void region (lines 9-11). If P_{rx} is found in buffer B_s, the hop count information is used to deal with the packet. The node discards the incoming packet from subset L or cancels forwarding the packet outside of the subset H (lines 14-20).

Once it has received a new packet, the node executes opportunistic coding. The packet from the void region is stored in buffer B_t to increase its lifetime. Then, the node attempts to

Algorithm 1 Forwarding With Opportunistic Coding

```
1:  Check $P_{rx}$ in buffers $B_s$ and $B_t$ and recorder $R_{discard}$
2:  if $P_{rx} \in R_{discard}$ then
3:      $R_{discard} \leftarrow P_{rx}$
4:  else if $P_{rx} \in B_t$ then
5:      if $hc(P_{rx}) < mHC$ then
6:          $B_t \leftarrow B_t - \{P_{rx}\}$
7:          $R_{discard} \leftarrow P_{rx}$
8:      else if $mHC \neq hc_{void}$ then
9:          $B_t \leftarrow B_t - \{P_{rx}\}$
10:         $B_s \leftarrow P_{rx}$
11:         set timer according to (19)
12:     end if
13: else if $P_{rx} \in B_s$ then
14:     if $hc(P_{rx}) \leq mHC$ then
15:         $B_s \leftarrow B_s - \{P_{rx}\}$
16:         $R_{discard} \leftarrow P_{rx}$
17:         cancel corresponding timer
18:     else
19:         $R_{discard} \leftarrow P_{rx}$
20:     end if
21: else
22:     if $hc(P_{rx}) = hc_{void}$ then
23:         $B_t \leftarrow P_{rx}$
24:     end if
25:     Set flag $F_{ec}$ according to (8)
26:     if $F_{ec} = TRUE$ then
27:         $P_{ec} = sliding\_window\_coding(B_s, B_t, P_{rx})$
28:         $B_s \leftarrow P_{ec}$
29:         set timer according to (19)
30:     else if $hc(P_{rx}) > mHC$ then
31:         $B_s \leftarrow P_{rx}$
32:         set timer according to (19)
33:     else
34:         $R_{discard} \leftarrow P_{rx}$
35:     end if
36: end if
```

encode the forward packet, and this procedure refers to (8) (lines 27-29). We define the coding probability p_{ec} as

$$p_{ec} = \begin{cases} 1, & hc_p = \infty; \\ min(1, \dfrac{\lambda_1}{N_L}), & N_L > \gamma_1, hc_i < hc_p < \infty; \\ min(1, \dfrac{\lambda_0}{max(N_S, 1)}), & N_L < \gamma_0, hc_p = hc_i; \\ min(1, \dfrac{\lambda_0}{max(N_H, 1)}), & (N_S + N_H) < \gamma_0, hc_p < hc_i; \\ 0, & otherwise. \end{cases} \tag{8}$$

where N_L, N_S and N_H are the numbers of neighboring nodes from subsets L, S, and H, respectively. hc_p is the hop count carried in the packet. γ_0 and γ_1 are the coding thresholds for nodes outside and inside subset L, respectively. λ_0 and λ_1 are the redundancy coefficients of coding for nodes outside and inside subset L, respectively.

In addition, CORS also executes opportunistic forwarding for the incoming packets from subset H (lines 31 and 32). Moreover, forwarding based on topology awareness is exploited when a node reconnects to the network. The main process is as follows: after forwarding a packet, the node discards the packet according to the topological information. If the node is in the void region or the network density around the node is very sparse, the packet is stored in buffer B_t to increase its lifetime. Each time a node reconnects to the network, it tries to forward the packets in buffer B_t with opportunistic coding.

C. SLIDING WINDOW-BASED CODING ALGORITHM

In the coding phase, each node uses the coding window decided by incoming packets and buffered packets to control the coding process. The coding window is defined as follows:

Definition 1: Considering a set of source nodes in a network and a series of sequence identifiers (SIDs) for the packets from each node, we define the subwindow W_{sub} as the sorting of the addresses of all active sensor nodes. Thus, the size of W_{sub} is N_{src}, which is the number of nodes in the network. We use an SID to mark each subwindow and use the ordering identifier (OID) of the node address to mark each element within a subwindow.

Definition 2: We define the sliding window W_{ec} as a set of subwindows marked with continuous SIDs. If we define the number of W_{sub} in a sliding window as N_{sW}. Thus, the number of elements in W_{ec} is $|W_{ec}| = N_{sW}N_{src}$. Then, W_{ec} can be represented as $W_{ec} = \{(SID_i, N_i)|i = 1, 2, \ldots, N_{sW}; j = 1, 2, \ldots, N_{src}\}$. On this basis, we use $P_i = \{(SID_{ik}, N_{ik})|k = 1, 2, \ldots\}$ to represent the ith coded packet P_i, where SID_{ik} and N_{ik} are the SID and the OID of the kth element in the coding vector of packet P_i, respectively.

The asynchronous generation of each packet and the multi-hop transmission in the network make it impossible for a node to receive the packets encoded within a small time interval. However, directly coding the packets with various insertion times results in large decoding delays. Considering this problem, we introduce a maximum buffering time τ_{buf} to filter the packets that are buffered locally. A packet with a buffering time larger than τ_{buf} is discarded and cannot participate in coding. This method can effectively limit the variations in the delays of the input packets and reduce coding overhead. Moreover, we set an interval for the encoding degree to control the encoding sparsity. Considering the above factors, we determine the condition of coding as below.

$$\begin{cases} maxSeq(P_{ec}, P_i) - minSeq(P_{ec}, P_i) \leq N_{sW}; \\ t_c - t_{insert}(P_i) \leq \tau_{buf}; \\ D_m \leq dim(V(P_{ec}) + V(P_i)) \leq D_M. \end{cases} \tag{9}$$

where P_{ec} is the input packet of the coding process in each round, P_i is the ith packet in the buffer, and t_c is the current time. $t_{insert}(P_i)$ and $V(P_i)$ are the insertion time and the

Algorithm 2 Sliding Window-Based Coding

```
1:  $P_{ec} \leftarrow P_0$
2:  for each packet $P_i \in (B_t \cup B_s)$ do
3:      if satisfies condition (9) then
4:          $W_{ec} \leftarrow P_{ec}, P_i$
5:          $G_{ec} \xleftarrow{W_{ec}} P_{ec}, P_i$
6:          $P_{ec} \xleftarrow{G_{ec}} P_{ec}, P_i$
7:      end if
8:  end for
9:  if $P_{ec} \in (B_t \cup B_s \cup R_{discard})$ then
10:     $P_{ec} \leftarrow P_0$
11: end if
```

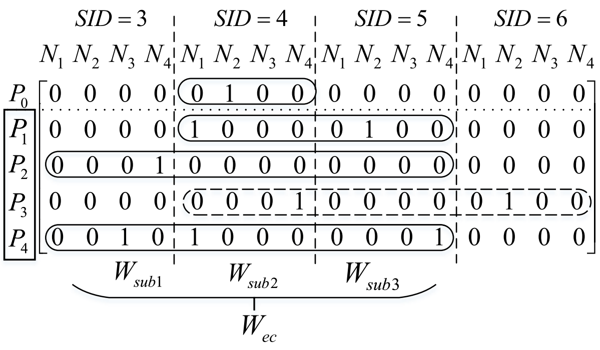

SID = 3
SID = 4
SID = 5
SID = 6
N1 N2 N3 N4
P0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
P1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0
P2 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
P3 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0
P4 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0
Wsub1
Wsub2
Wsub3
Wec

FIGURE 3. An example of sliding window-based coding.

coding vector of packet P_i, respectively. D_m and D_M are the minimum and maximum dimensions of a coded packet P_{ec}, respectively.

Algorithm 2 shows the process of sliding window-based coding. First, we set packet P_0 that is currently received to the coding output P_{ec}. Then, we choose the valid packets in buffers B_t and B_s in succession to execute the encoding process. In each round, we update the encoding window W_{ec} by using the coded packet of the last round together with the input packet P_i in the current round (line 4). On this basis, G_{ec} is mapped according to W_{ec} and is further used to encode packets P_i and P_{ec} (lines 5-6). This process continues until the degree of the coding vector reaches the maximum value D_M or all the valid packets in each buffer are chosen. In this way, we obtain P_{ec} with window coding by using all the potential packets in the buffers. Last, we set P_0 to P_{ec} if the coded packet P_{ec} has came across before (line 10).

We use an example to explain the sliding window-based coding algorithm. As shown in Fig. 3, we assume that the number of elements in W_{sub} is 4 and that each coding window W_{ec} has 3 subwindows. For simplicity, we consider encoding over a Galois field GF(2). The current node buffers four packets: $P_1 = \{(4, 1), (5, 2)\}$, $P_2 = \{(3, 4)\}$, $P_3 = \{(4, 4), (6, 2)\}$, and $P_4 = \{(3, 3), (4, 1), (5, 4)\}$. After receiving packet P_0, we successively attempt to encode with the packets in each buffer. As shown in Fig. 3, $P_0 = \{(4, 2)\}$, such that the SID of the potential coding window for P_0 ranges from 2 to 6 since only one SID in this packet is 4. Therefore, we can find that P_1 is inside of this interval. Then, we encode P_1 with P_0

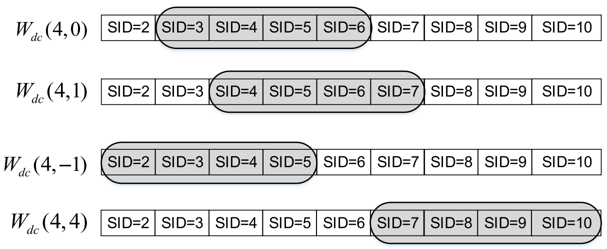


FIGURE 4. Sliding window for decoding.

and obtain a coding output P_{ec}, which can be represented as $P_{ec} = \{(4, 1), (4, 2), (5, 2)\}$. Next, we obtain a new coding result $P_{ec} = \{(3, 4), (4, 1), (4, 2), (5, 2)\}$ by encoding packets P_2 and P_{ec}. The encoding window with a value of $W_{ec} = \{(i, N_j)|i = 3, 4, 5, j = 1, 2, 3, 4\}$ is unchangeable since the number of subwindows reaches the maximum value. Then, the encoding process finishes when $D_M = 4$. If we set a larger value for D_M, it attempts to encode with P_3. In the next round, however, P_3 is obviously outside of W_{ec}, as it has the packet with the coding information of $SID = 6$. Thus, the node bypasses P_3 and attempts to encode with P_4. This process continues until the degree of the coding packet reaches the maximum value D_M.

D. SLIDING WINDOW-BASED DECODING ALGORITHM

The base station use sliding window-based decoding (SWD) algorithm to decode with the accumulated information. The sliding window for decoding is defined as below.

Definition 3: We define the current decoding window W^0_{dc} as a set of subwindows marked with continuous SIDs. If we define the number of subwindows W_{sub} in a decoding window as N_{sDW}, then, the number of elements in W^0_{dc} is $|W_{dc}| = N_{sDW}N_{src}$. Moreover, W^0_{dc} can be represented as $W^0_{dc} = \{(SID_i, N_i)|i = 1, 2, \ldots, N_{sDW}; j = 1, 2, \ldots, N_{src}\}$. On this basis, we use $W_{dc}(L, r) = \{(SID_i+r, N_i)|i = 1, 2, \ldots, L; j = 1, 2, \ldots, N_{src}\}$ to represent an arbitrary decoding window, where $L = N_{sDW}$ and r is the number of subwindows to the right of window W^0_{dc}.

From the definition above, we can obtain $W_{dc}(L, 0) = W^0_{dc}$. As shown in Fig. 4, when $r > 0$, the new decoding window is on the right side of window W^0_{dc}. When $r < 0$, the new decoding window is on the left side of window W^0_{dc}.

Definition 4: We use the function $Seq()$ to obtain the set of SIDs in a packet or in a sliding coding/decoding window. On this basis, we use $Seq(A) \subseteq Seq(B)$ to represent that the set of SIDs in A is inside of that in B. In contrast, we use $Seq(A) \not\subset Seq(B)$ to represent that the set of SIDs in A is outside of that in B. For instance, as shown in Fig. 4, we assume that the decoding window is $W^0_{dc} = \{(sid, N_i)|i = 3, 4, 5, 6; j = 1, 2, \ldots, N_{src}\}$. Then, coding window $W_1 = \{(sid, N_i)|i = 4, 5, 6; j = 1, 2, \ldots, N_{src}\}$ is inside of decoding window W^0_{dc}. Coding window $W_2 = \{(sid, N_i)|i = 2, 3; j = 1, 2, \ldots, N_{src}\}$ is outside of W^0_{dc} since SID = 2 is not in the set of SIDs in W^0_{dc}.

Algorithm 3 Sliding Window-Based Decoding

```
1: Obtain encoding information from $P_{rx}$
2: if $maxSeq(P_{rx}) > maxSeq(W_{dc})$ then
3:     $N_{rW} \leftarrow N_{rW} + 1$
4: end if
5: if $Seq(P_{rx}) \subseteq Seq(W_{dc})$ then
6:     $G_{dc} \overset{W_{dc}(l,0)}{\longleftarrow} P_{rx}$
7:     $gaussian\_elimination(G_{dc})$
8: else if $(N_{rW}/N_{DW}) > TH_M$ then
9:     $W_{dc}(l, 0) = W_{dc}(l, 1)$
10:    $G_{dc} \overset{W_{dc}(l,0)}{\longleftarrow} G_{dc}, M_{dc}, M_{ndc}, P_{rx}$
11:    $gaussian\_elimination(G_{dc})$
12: else
13:    if $maxSeq(P_{rx}) > maxSeq(W_{dc})$ then
14:        $W_{tmp} = W_{dc}(l, maxSeq(P_{rx}) - maxSeq(W_{dc}))$
15:    else if $minSeq(P_{rx}) < minSeq(W_{dc})$ then
16:        $W_{tmp} = W_{dc}(l, minSeq(P_{rx}) - minSeq(W_{dc}))$
17:    end if
18:    $G_{tmp} \overset{W_{tmp}}{\longleftarrow} G_{dc}, M_{dc}, M_{ndc}, P_{rx}$
19:    $gaussian\_elimination(G_{tmp})$
20: end if
21: Record decoding results and update $G_{dc}, M_{dc}, M_{ndc}$
```

After arriving at the sink node, each packet is further relayed to the base station via an ocean satellite. The base station discards the packet successfully received before. Otherwise, it executes the sliding window-based decoding algorithm to decode the original information. Two storage regions are occupied for the decoding process: memory M_{dc} is used for storing decoding information, and memory M_{ndc} is used for storing the encoding information of the packets that are expected to be decoded. In addition, the decoding matrix G_{dc} for the current window is buffered and is updated together with the sliding decoding window W^0_{dc}.

Once additional incoming packets with coding windows on the right side of the current decoding window appear, W^0_{dc} should slide to the right. We use a moving threshold TH_M (ranging from 0 to 1) to control the movement of the current decoding window. N_{rW} represents the number of packets whose maximum SIDs are larger than the maximum SID of the decoding window. When the ratio of packets on the right side of the current decoding window to the decoding window size N_{rW}/N_{DW} is higher than TH_M, the decoding window slides to the right with a unit of one subwindow.

Algorithm 3 depicts the pseudocode of the sliding window-based decoding algorithm. Once a new packet P_{rx} is received, the base station obtains the coding information from it (line 1). Then, it increases N_{rW} if the maximum SID of P_{rx} is larger than that of the current decoding window (lines 2-4). Thereafter, if the coding window of P_{rx} is inside of the current decoding window W^0_{dc}, it creates a new row in the decoding matrix and maps the coding vector of P_{rx} to this row (line 6). Otherwise, if the moving condition is satisfied, the decoding window slides to the right and the decoding

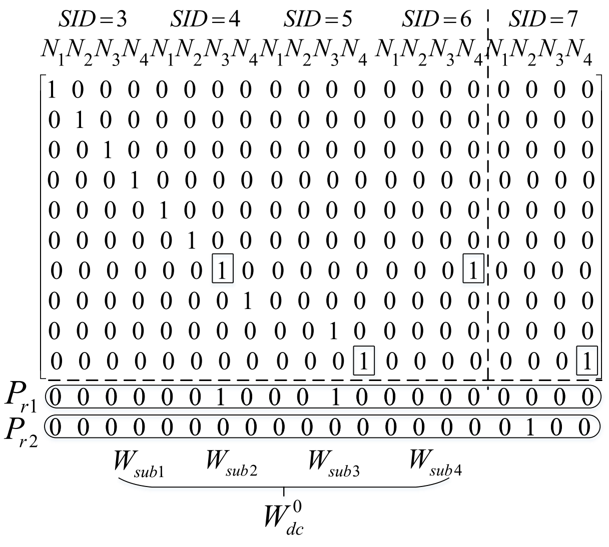


FIGURE 5. An example of sliding window-based decoding.

matrix is updated with the information in memory M_{dc} and memory M_{ndc} and packet P_{rx} (lines 9-10): the base station refreshes the decoding matrix according to the new decoding window; then, it maps the coding information of memory M_{dc} and memory M_{ndc} and the coding vector of P_{rx} to the decoding matrix. If the decoding window fails to slide, the base station constructs a temporary decoding matrix G_{tmp} based on G_{dc}, M_{dc}, B_{dc}, and P_{rx} (lines 13-18). Finally, the base station executes Gaussian elimination to decode the original information and record the corresponding results (lines 7, 11, 19, 21) [18].

We use an example to explain the sliding window-based decoding algorithm. As shown in Fig. 5, use GF(2). Generally, the decoding matrix G_{dc} is a sparse matrix with fewer than 1 element. The current decoding window, represented as $W^0_{dc} = \{(sid, N_j)|sid = 3, 4, 5, 6; j = 1, 2, 3, 4\}$, has $L = 4$ subwindows. The number of elements in each subwindow is 4. Once packet $P_{r1} = \{(4, 3), (5, 3)\}$ is received, the base station obtains a coding window $W_{ec1} = \{(sid, N_j)|sid = 4, 5; j = 1, 2, 3, 4\}$ and compares it with window W^0_{dc}. Since $W_{ec1} \subseteq W^0_{dc}$, the base station adds the coding vector of packet P_{r1} to the decoding matrix G_{dc}. Then, it decodes out the original packets $\{(4, 3)\}$ and $\{(6, 4)\}$ by using Gaussian elimination. After receiving packet $P_{r2} = \{(7, 2)\}$, the node increase N_{rW} to 2 since there already has one packet with a larger SID than that of current decoding window. Then, the decoding window moves right according to $W_{dc}(4, 1) = \{(sid, N_j)|sid = 4, 5, 6, 7; j = 1, 2, 3, 4\}$ since $N_{rW}/N_{DW} = 0.125$ is higher than the preset $TH_M = 0.1$. Once a packet with a coding window outside of the current decoding window is obtained, the base station constructs a temporary matrix. For instance, Upon receiving packet $P_{r3} = \{(2, 4), (4, 1)\}$, the base station constructs a temporary decoding matrix G_{tmp} according to a temporary decoding window $W_{tmp} = \{(sid, N_j)|sid = 2, 3, 4, 5; j = 1, 2, 3, 4\}$, which is obtained by sliding W^0_{dc} left with a unit of one subwindow.

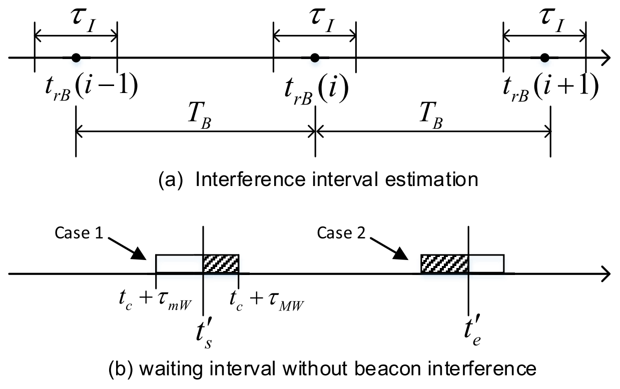


FIGURE 6. Interference interval estimation and backoff.

Then, the matrix G_{tmp} is used for decoding by using Gaussian elimination.

E. BACKOFF STRATEGY

Each sink periodically establishes its topological information by broadcasting beacon messages. The control messages may occupy the network channel for a period of time. Thus, the data packets generated in these time intervals may interfere with the control messages. To address this problem, we introduce a backoff strategy to set the timer for sending each data packet by estimating the interference interval. First, each node calculates the waiting time interval of each packet by using available topological information and depth information. Then, the latest update time is used to estimate the times of subsequent updates. Finally, the waiting time interval is calculated according to the estimated interference interval in the previous step.

We use the topological information and depth information of a node to calculate the minimum waiting time as follows:

$$\tau_{mW} = max(max(2 + hc_i - hc_j, 0) \cdot \tau_p - \tau_v, 0) \tag{10}$$

where hc_i and hc_j are the hop counts of the current node i and last forwarder j, respectively. τ_p is the maximum propagation delay of the network, while τ_v is the propagation delay in the vertical direction between the forwarder and the current node; this can be calculated as

$$\tau_v = \frac{d_j - d_i}{C} \tag{11}$$

where d_i and d_j are the depths of the current node i and last forwarder j, respectively. C is the average propagation speed of sound in the sea area. Then, we calculate the maximum waiting time τ_{MW} as

$$\tau_{MW} = \tau_{mW} + min(DG, CW) \cdot \tau_D \tag{12}$$

where DG is the topology degree of the current node, CW is the size of the competing window, and τ_D is the transmission delay of the data packet.

As shown in Fig. 6a, each node estimates the subsequent receiving times of beacons by using the receiving time of the latest beacon. In this way, the time required to receive the ith beacon packet can be estimated as

$$t_{rB}(i) = t_{rB}(0) + i \cdot T_B, \quad i = 1, 2, \ldots \tag{13}$$

where $t_{rB}(0)$ is the receiving time of the latest beacon.

We consider multihop interference to be the longest interference distance in the underwater environment [34]. We calculate the interference interval with the possible receiving time of each beacon received in the multihop nodes. The start time for the farthest node in subset L to receive beacon $t_s(i)$ and the end time for the farthest node in subset H to finish receiving beacon $t_e(i)$ are represented as

$$\begin{cases} t_s(i) = t_{rB}(0) - (\sum_{j=1}^{M} \tau_p(j) + (M+1)\tau_B) + i \cdot T_B - (\tau'_p + \tau_D); \\ t_e(i) = t_{rB}(0) + (\sum_{j=1}^{M} \tau_p(j) + M\tau_B) + i \cdot T_B - \tau'_p + \tau_a; \\ i = 1, 2, \ldots \end{cases} \tag{14}$$

where τ_B is the transmission delay of a given beacon. M is the maximum difference of hop count between interference nodes and reference node. τ'_p is the direct propagation delay of the farthest interference node, respectively. $\tau_p(j)$ is the propagation delay of the jth hop on the path to the farthest interference node. τ_a is the adjustment time. The maximum interference range R_I can be represented as $R_I = \gamma' R_{tx}$, where γ' is the spatial reuse factor and R_{tx} is the transmission range [34]. Then, we have $\tau'_p = \gamma' \tau_{p,Max}$. Furthermore, we set $\tau_p(j)$ with the maximum propagation delay of one hop $\tau_{p,Max}$. Then, we rewrite (14) as below.

$$\begin{cases} t_s(i) = t_{rB}(0) - (M + \gamma')\tau_{p,Max} + i \cdot T_B - (M+1)\tau_B - \tau_D; \\ t_e(i) = t_{rB}(0) + (M - \gamma')\tau_{p,Max} + i \cdot T_B + M\tau_B + \tau_a; \\ i = 1, 2, \ldots \end{cases} \tag{15}$$

Then, the interference interval of topology establishment $[t'_s, t'_e]$ can be estimated as

$$\begin{cases} t'_s = t_s(i); \\ t'_e = t_e(i); \\ t_e(i-1) \le t_c < t_e(i), \quad i = 1, 2, \ldots \end{cases} \tag{16}$$

As shown in Fig. 6b, referring to (16), we estimate the valid minimum waiting time interval τ'_{mW} and the valid maximum waiting time interval τ'_{MW} as follows:

$$\tau'_{mW} = \begin{cases} t'_e - t_c, & \tau_{mW} < (t'_e - t_c) < \tau_{MW}; \\ \tau_{mW}, & otherwise. \end{cases} \tag{17}$$

$$\tau'_{MW} = \begin{cases} t'_s - t_c, & \tau_{mW} < (t'_s - t_c) < \tau_{MW}; \\ \tau_{MW}, & otherwise. \end{cases} \tag{18}$$

At last, the waiting time τ_W is calculated as

$$\tau_W = \tau'_{mW} + uniform(\tau'_{MW} - \tau'_{mW}) \tag{19}$$

V. PROTOCOL ANALYSIS

In this section, we analyze the reliability and effectiveness of the algorithm in our design. We first compare the candidate set selection of CORS with those of typical OR protocols. Then, we analyze the coding strategy used in our design. Finally, we compare the SWC algorithm with random linear coding (RLC) in terms of coding overhead.

A. CANDIDATE SET COMPARISON

We compare the candidate sets of the CORS protocol with those of typical hop count-based protocols (IVAR and DOVR) and a typical pressure-based protocol (DBR).

In the IVAR and DOVR protocols, the candidate set for node i includes the nodes in subset L within the transmission range, which can be represented as

$$C_{DOVR}(i) = C_{IVAR}(i) = \{N_j | j \neq i, d_{ij} < R_{tx}, hc_i > hc_j\} \quad (20)$$

where N_j is the jth node in subset L, while d_{ij} is the distance between node i and node j. hc_i and hc_j are the hop counts of reference node i and node j, respectively.

In CORS, the candidate set of node i contains all the nodes in subset L within the transmission range. In addition, the nodes outside of subset L are included in the candidate set with probability. The candidate set of CORS is

$$C_{CORS}(i) = \{N_j | j \neq i, d_{ij} < R_{tx}, p_j < p_{fw}\} \quad (21)$$

where p_{fw} is the probability of forwarding a packet. The CORS protocol contains two forwarding mechanisms: forwarding directly and forwarding with opportunistic coding. p_{fw} can be calculated as

$$p_{fw} = p_{dr} + p_{ec} \quad (22)$$

where p_{ec} (calculated in (8)) is the probability of forwarding with coding, while p_{dr} is the probability of forwarding directly, which is given as

$$p_{dr} = \begin{cases} 1 - p_{ec}, & hc_p > hc_i; \\ 0, & hc_p \leq hc_i. \end{cases} \quad (23)$$

where hc_p is the hop count of incoming packet.

Referring to (20) and (21), we can obtain

$$C_{DOVR}(i) \subseteq C_{CORS}(i) \quad (24)$$

If and only if $\{N_j | j \neq i, d_{ij} < R_{tx}, hc_i \leq hc_j\} = \varnothing$, we have $C_{IVAR}(i) = C_{DOVR}(i) = C_{CORS}(i)$. Therefore, the CORS protocol can effectively expands the candidate set compared with those of typical hop count-based protocols.

In DBR, the candidate set of each node is represented as

$$C_{DBR}(i) = \{N_j | j \neq i, d_{ij} < R_{tx}, (h_i - h_j) > h_{TH}\} \quad (25)$$

where h_i represents the depth information of node i, while h_{TH} is the depth threshold used in DBR, which ranges from $[-R_{tx}, R_{tx}]$. When $h_{TH} = -R_{tx}$, any node that receives the packet can participate in competitive forwarding. In this case, the DBR protocol is equivalent to the flood protocol. In the case where $h_{TH} = R_{tx}$, the DBR protocol fails to forward packets as the candidate set $C_{DBR}(i) = \varnothing$.

Referring to (25), we can find that the candidate set of DBR may cover the nodes in the void region. However, the nodes in the void region cannot reliably forward packets, and this heavily influences the delivery performance of the network in sparse scenarios. In CORS, the nodes in the candidate set are more reliable than those in DBR, as the hop count information is periodically updated. We can improve the reliability of the candidate set in DBR by decreasing the parameter h_{TH}. However, additional redundant transmissions are introduced by setting a small h_{TH}, and this may result in a significant influence on the effectiveness of the DBR protocol. Therefore, the CORS protocol can more effectively optimize the candidate set than the DBR protocol.

B. OPPORTUNISTIC CODING STRATEGY

Referring to (8), we can find that γ_0 can be used to control the coding candidate set of the nodes outside subset L. The nodes in subset S and in the transmission range may participate in opportunistic coding in the case where $N_L < \gamma_0$. The nodes in subset H in the transmission range may participate in coding in the case where $(N_L + N_S) < \gamma_0$. When γ_0 is small, the nodes in subsets H and S can participate in the FOC strategy in the case where there are few nodes in subset L. This is especially true when $\gamma_0 = 1$, as only the nodes in the void region satisfy the coding condition. In contrast, we can enlarge the selection range of the candidate set with the nodes outside subset L by increasing γ_0.

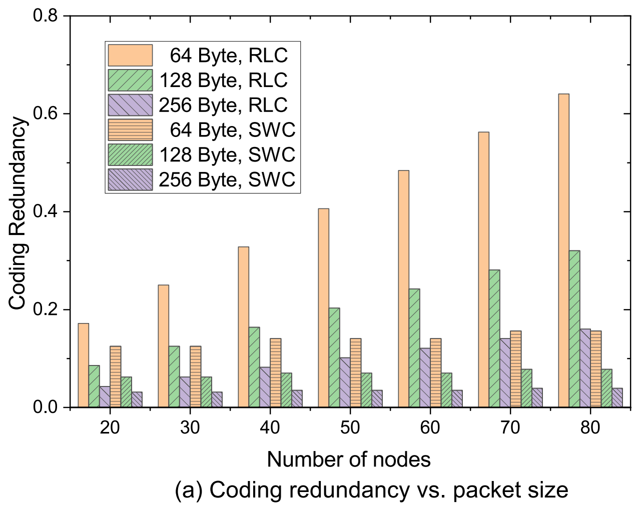
The parameter γ_1 is used for controlling the coding of the nodes in subset L. When γ_1 is small, the nodes in subset L participate in coding with a certain probability. In the case where $\gamma_1 = 0$, all the nodes in subset L execute the FOC strategy. However, when γ_1 is large enough, all the nodes in subset L forward the packet without coding.

The parameter λ_0 controls the probability of dividing the candidate set with nodes outside subset L. With small values of λ_0, a node outside of subset L rarely participates in the FOC strategy, especially in sparse scenarios. In contrast, when λ_0 is set with a larger value, the candidate set includes more nodes outside of subset L. However, this may cause local flooding in the network.

The parameter λ_1 controls the probability of dividing the candidate set with the nodes in subset L. In cases with small values of λ_1, the nodes in subset L rarely participate in the FOC strategy. In cases with larger values of λ_1, nearly all the nodes in subset L can participate in the FOC strategy. However, high variation in the coded packets may lead to transmission storms, which may result in sharp declines in network performance.

C. SLIDING WINDOW-BASED CODING ALGORITHM

Since the number of sensor nodes is a constant value, the size of the sliding window depends on the number of subwindows N_{sW}. The choice of N_{sW} is related to the traffic rate. At a low traffic rate, the difference between the SIDs



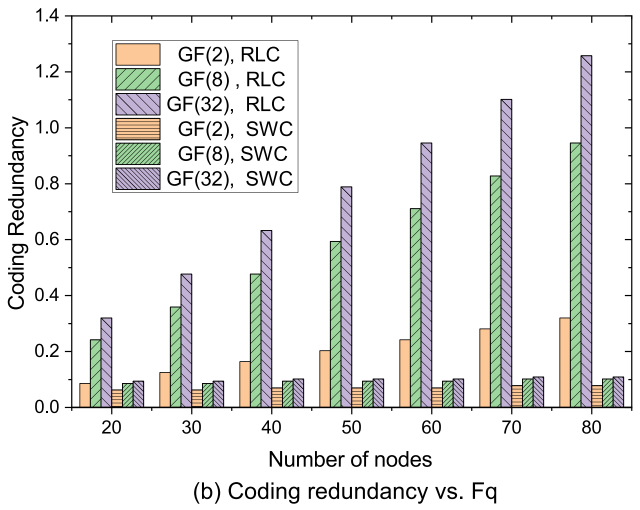


FIGURE 7. **Effects of various parameters on coding redundancy.**

received by the packets within time interval τ_{buf} is small, and thus, the number of input packets that require coding is slightly related to N_{sW}. In contrast, the probability of coding can be improved by increasing N_{sW} at high traffic rates since the buffered packets have various SIDs in time interval τ_{buf}.

In addition, the coding performance is related to the sparsity of network coding. The decoding performance of random linear coding (RLC) has a positive correlation with the average coding degree in cases with small coding degrees. RLC can achieve the best performance when the coding degree equals half of the size of the input encoding set [35], [36]. The coding vector is conveyed by an encoding packet since it is generated randomly in each node. The length of the encoding vector increases as the network size increases. However, a packet with a large size results in a large packet loss rate and has a high probability of being interfered with by other nodes.

In RLC, the coding overhead is

$$R_{rlc} = |W_{ec}| \cdot q + L_{SID} \tag{26}$$

where 2^q is the size of Galois field $GF(2^q)$ and L_{SID} is the size of the SID in the encoding window.

In multihop UWSNs, the coding vector is naturally sparse since the source node of each packet is separated topologically. Considering this characteristic, we reduce the coding overhead by limiting the maximum coding degree to some extent. Thus, the coding vector can be represented by only recording the position of each nonzero value. The coding overhead in CORS is given by

$$R_{swc} = D_M \cdot (\lceil log_2(N_{src}) \rceil + \lceil log_2(N_{sW}) \rceil + q) + L_{SID} \tag{27}$$

We set the subwindow size to 4 and then compare the coding redundancies of these two coding algorithms in various scenarios. Fig. 7a shows that the coding redundancies of these two coding algorithms vary with the network size and the packet size when we use $GF(2)$. The coding redundancy of RLC is high and significantly increases as the network size becomes larger. In contrast, the coding redundancy of the SWC algorithm is obviously smaller than that of RLC and changes slightly with different network sizes. Fig. 7b shows that the coding redundancies of these two coding algorithms vary with the network size and q when we set the packet size to 128 bytes. We can observe that the coding redundancy of the SWC algorithm is obviously smaller than that of RLC for different values of q. In addition, the coding redundancy of the SWC algorithm is slightly influenced by q as the network size increases.

VI. PERFORMANCE EVALUATION

A. SIMULATION SCENARIOS AND SETTINGS

In this section, we evaluate the performance of the proposed CORS protocol by using OPNET software, and we compare it with the DBR, IVAR, and DOVR protocols. We consider a sparse underwater wireless sensor network with its number of nodes ranging from 20 to 80. In each scenario, all the source nodes are randomly deployed in a region with a size of $500m \times 500m \times 500m$, where four sink nodes are deployed on the surface of the water to collect and relay data. Considering the seasonal characteristics of this sea area, we set the wind speed to 6.5 m/s and the ship level to 0.5. σ^2 ranges from 0.0 to 1.0 dB [37], [38]. In addition, an extended 3D version of the meandering current mobility (MCM), which considers the effect of meandering subsurface currents (or jet streams) and vortices, is adopted to simulate the mobility of nodes [3]. As considered in [10], the main jet speed is set to 0.3 m/s.

We assume that each node is equipped with a high-speed mini-modem S2CM-HS that can offer incredible 62.5 kbps

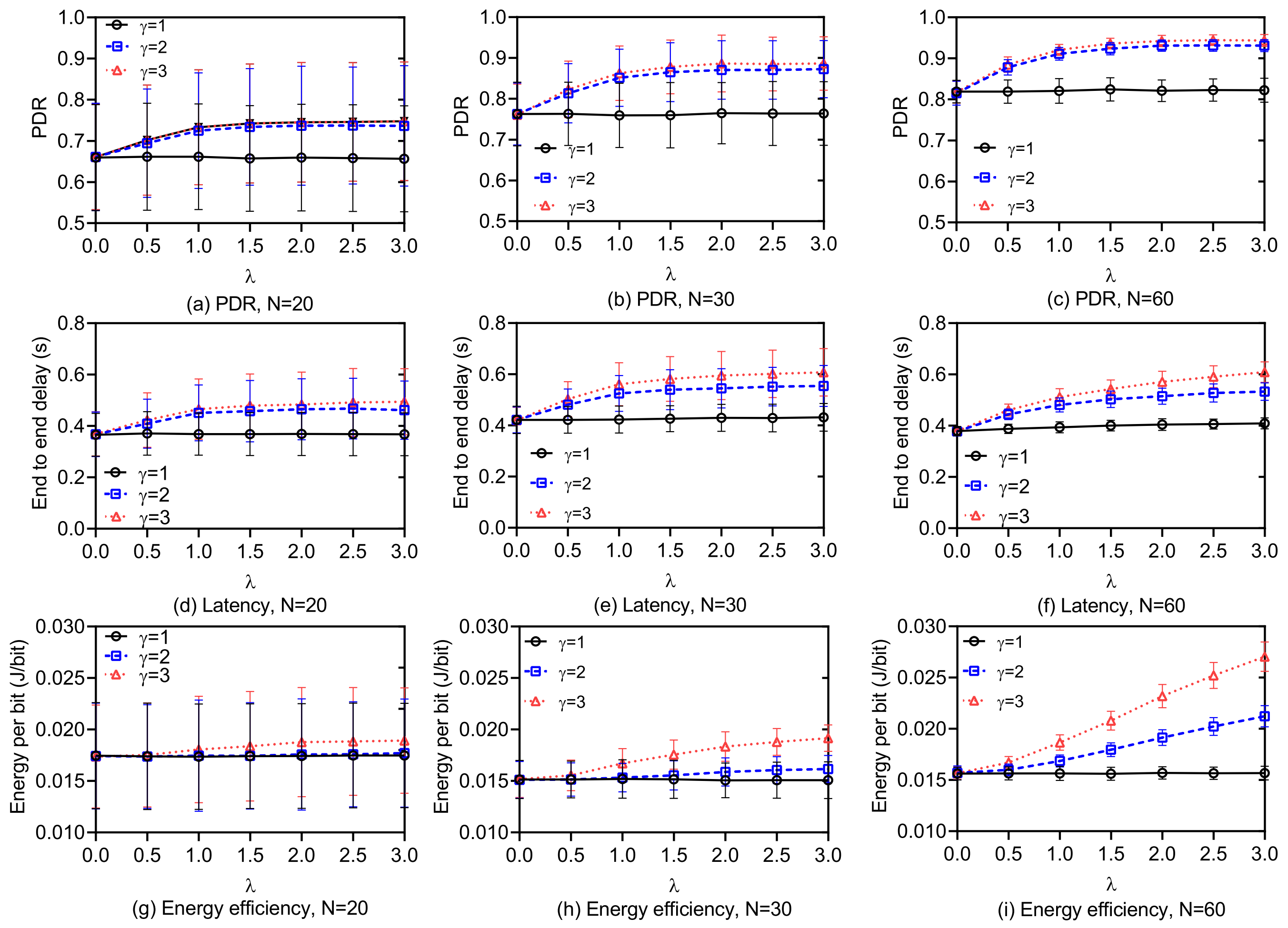


FIGURE 8. Comparisons of network performances with various values of λ and γ in networks of different densities.

for short-range transmission [39]. We choose a frequency band of 120-170 kHz. The data rate is set to 50 kbps, as used in [10], [27]; thus, the bandwidth efficiency is 1 bps/Hz. We set the source level to 134.6 dB *re* uPa @ 1 m and consider the SINR threshold to be 10 dB, which results in an average transmission range of $R_{tx} = 180$ m. To evaluate the energy efficiency, we assume the transmission/receiving/idle power to be 3.5/0.8/0.08 W. At the medium access control (MAC) layer, we use the carrier sense multiple access (CSMA) protocol since it can avoid collisions to some extent [10]. We assume that the overhead of the physical layer and the MAC layer is 4/3 byte. In the network layer, the overhead of the control packets for three hop count-based protocols is 2 bytes. The data packet size is set to 128 byte. In addition, we consider the overhead of data packets with sizes of 2/2/1/4 bytes for DBR/IVAR/DOVR/CORS. Considering the energy constrained node with limited hardware capabilities in this UWSN, we use GF(2) since the network coding process can be accomplished by XOR operation in GF(2), and which has the least coding overhead and decoding overhead. Thus, the coding overhead of the coded packet in CORS is calculated by referring to (27). We use depth threshold $h_{th} = 0$ m and parameter $\delta_{DBR} = R_{tx}/2$ in the DBR protocol. We set $CW = 6$ and $\gamma' = 2$ in CORS. The beacon interval T_B is set to 20 s for all hop count-based protocols. All the data packets are generated according to a Poisson process with three levels of traffic rates: $\{0.1, 0.5, 1.0\}$ packets/min. The simulation time of each run lasts 3600 s. The average value together with the standard deviation of 50 runs in different scenarios are used to demonstrate the simulation results. We use three performance metrics to reflect the network performances of different protocols, and these metrics are defined as follows.

1) PACKET DELIVERY RATIO (PDR)

The ratio of packets successfully received by any sink node to the packets sent by source nodes.

2) END-TO-END DELAY

The time required for a packet to arrive at any sink plus the generation time of the packet.

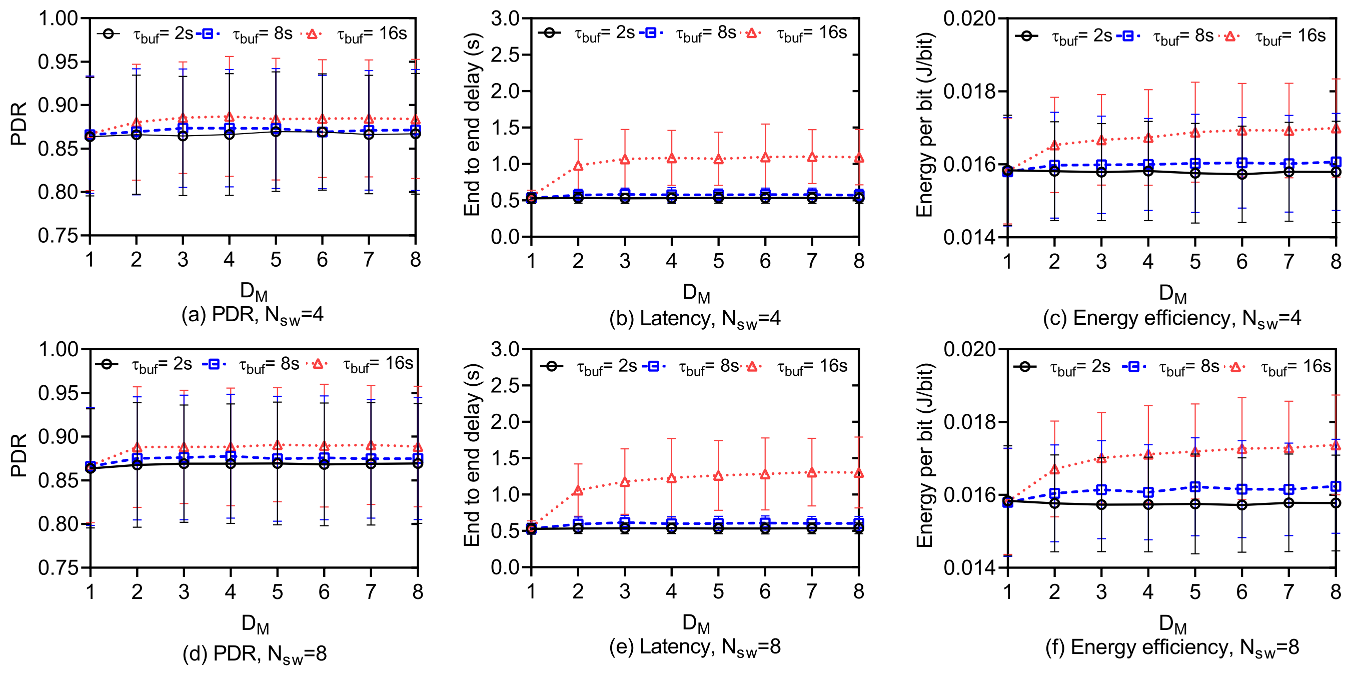


FIGURE 9. **Comparisons of network performances with various values of D_M and τ_{buf} when $N_{SW} = 4, 8$.**

3) ENERGY PER BIT

The energy consumption of the network for successfully delivering a bit of data to the sinks.

B. SIMULATION RESULTS

We first study the impacts of several parameters in CORS, and then we evaluate CORS against DBR, IVAR and DOVR in terms of the PDR, network latency, and energy efficiency.

1) IMPACT OF CODING PARAMETERS

For simplicity, we set $\lambda_1 = \lambda_0 = \lambda$ and $\gamma_1 = \gamma_0 = \gamma$, and we also consider the traffic rate $\lambda_T = 0.5$ packets/min and channel variance $\sigma^2 = 0.5$ dB. Then, we simulate network performances with different values of λ and γ. Fig. 8 shows that the PDR, end-to-end delay and energy per bit vary with various values of λ and γ in networks with different densities. As shown in Fig. 8, the PDR performances of all the protocols increase as the value of γ is increased. Furthermore, network latency and energy consumption also increase as the value of γ increases. This is because a large γ relaxes the restriction on the encoding threshold, which in turn enlarge the candidate set. In particular, the PDR significantly increases as γ is changed from 1 to 2. When $\gamma = 1$, only the nodes in the void region can be incorporated into the candidate set. As $\gamma > 1$, additional nodes in sparse regions can be incorporated into the candidate set. However, more coded packets can result in higher decoding delays, which in turn lead to higher latency and higher energy consumption.

As shown in Fig. 8, the PDR is slightly improved as λ becomes large in the case where $\gamma = 1$ and is significantly improved with larger values of λ in the case where $\gamma > 1$. Referring to (8), only the neighbors of the void node can participate in encoding in the case where $\gamma = 1$. Since a small proportion of the nodes are in the void region, increasing the probability of coding for nodes in the void region has little impact on the network performance. In cases where $\gamma > 1$, additional nodes from various subsets are put into the candidate set. In this case, increasing the probability of coding results in more nodes participating in forwarding, which can significantly improve the reliability of transmission. After λ reaches a certain value, the PDR performance is slightly improved, but the network latency changes significantly as λ is increased. This is mainly because the coding gain reaches the maximum value with a certain level of coding redundancy. At this moment, introducing more nodes to participate in coding would lead to broadcast storm problems and therefore increase network congestion. In addition, the backoff strategy used in CSMA may introduce much network latency.

In scenarios with 20 sensor nodes, the network is extremely sparse, and the number of coded packets increases slightly, which results in a smaller delay variation with different values of γ. In contrast, when deploying 60 sensor nodes in the same region, the network congestion present can obviously influence the network latency. Furthermore, more encoding redundancy can significantly increase the energy overhead, which results in lower energy efficiency. As shown in Fig. 8, the variance of the network performances in sparse scenarios is larger than in dense scenarios. This is because the nodes sparsely deployed in these scenarios result in a larger variance for the network topology, which in turn leads to a large variance with regard to the network performances in different scenarios.

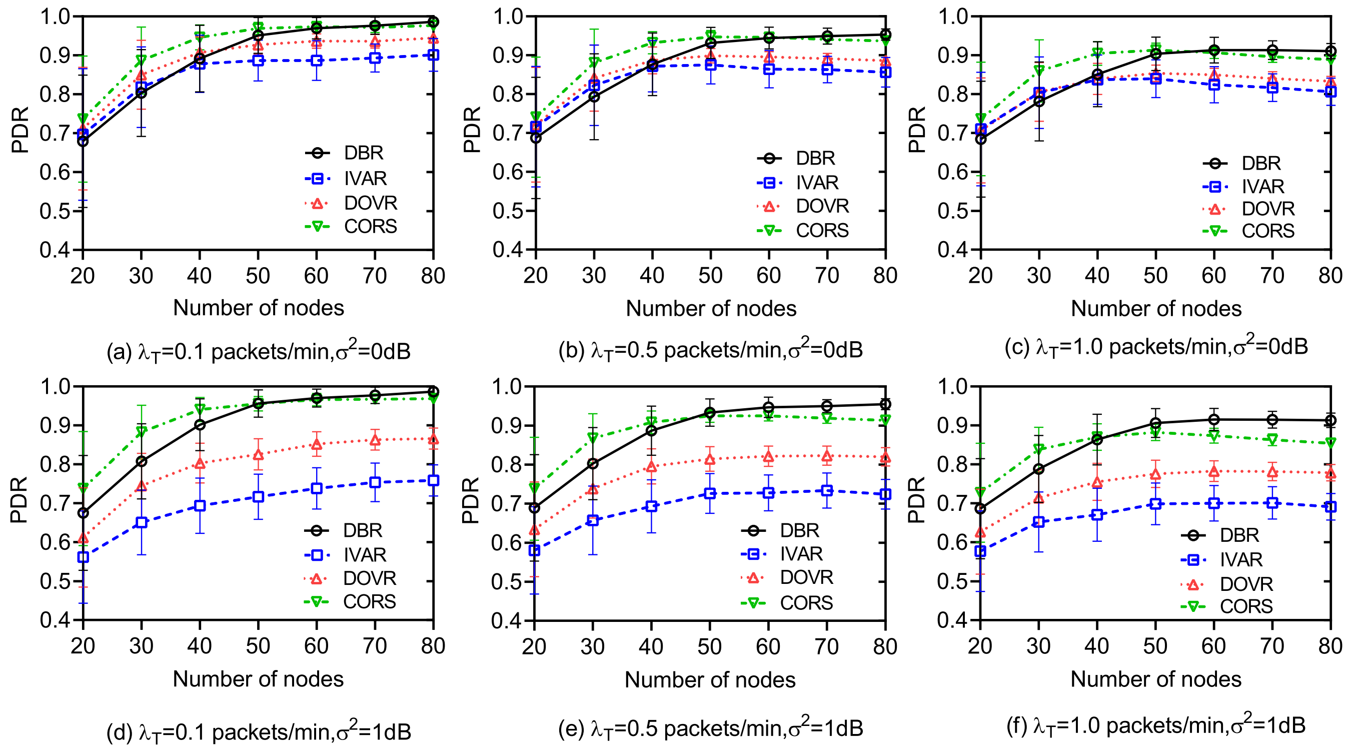


FIGURE 10. **Comparisons of PDR performances in networks with different densities.**

Fig. 9 shows that the PDR, end-to-end delay and energy per bit vary with various values of D_M, τ_{buf}, and N_{sW} when the number of nodes is 30. The PDR increases as N_{sW} increases. This is because a larger N_{sW} expands the interval of encoding symbols and then increases the coding dimension. However, increasing the proportion of coded packets results in higher network latency. Furthermore, the decreasing correlation of the encoding results between each pair of neighbors leads to high coding redundancy, which introduces unnecessary energy consumption.

The PDR increases with increasing network latency and energy consumption as τ_{buf} increases. This is because more valid input packets can be obtained in a longer buffer time interval. Thus, an increase in the probability of coding can effectively improve the PDR. However, the network latency increases significantly with increasing buffer time interval.

The PDR increases significantly as D_M increases at lower values and varies slightly as D_M further increases. CORS adopts opportunistic forwarding without coding in the case where $D_M = 1$. When $D_M > 1$, network coding can improve the reliability of network transmission. However, due to the regional correlation between the packets in each pair of nodes within adjacent regions, the coding gain reaches its maximum value and cannot be further improved by increasing D_M.

2) COMPARISON WITH DIFFERENT PROTOCOLS

Fig. 10 shows the PDRs of four protocols for various network sizes at different traffic rates. In sparse scenarios, all four protocols experience degraded PDR performances since the candidate sets of these protocols are small and even void in some regions. As expected, CORS performs better than the other three protocols. This is due to the reasons below. DBR is heavily influenced by the void problem, as candidate set selection is based on depth information, which in turn leads to a local optimization problem. In contrast, hop count-based protocols can alleviate the void problem by using hop counts. In addition, CORS extends the candidate set and incorporates opportunistic coding, which can effectively improve the delivery performance of a given network in sparse scenarios.

As the network size continually increases, the PDRs of all four protocols significantly increase and then remain stable. This is because the network connectivity is improved as the network density increases. At this time, the main factors that affect forwarding are collisions and multihop interference. For a traffic rate of $\lambda_T = 1.0$ packets/min, the collisions in the network increase heavily, which leads to serious performance degradation compared to that produced with $\lambda_T = 0.1$ packets/min. We can also observe that the PDR performance of DBR improves significantly compared with those of the other protocols. At a higher network density, the performance of DBR is even better than those of the other three protocols. This is because the collisions and interference introduced by the control message increase heavily at large network sizes in hop count-based protocols. Nevertheless, CORS outperforms the other two hop count-based protocols. First, the interference avoidance mechanism introduced

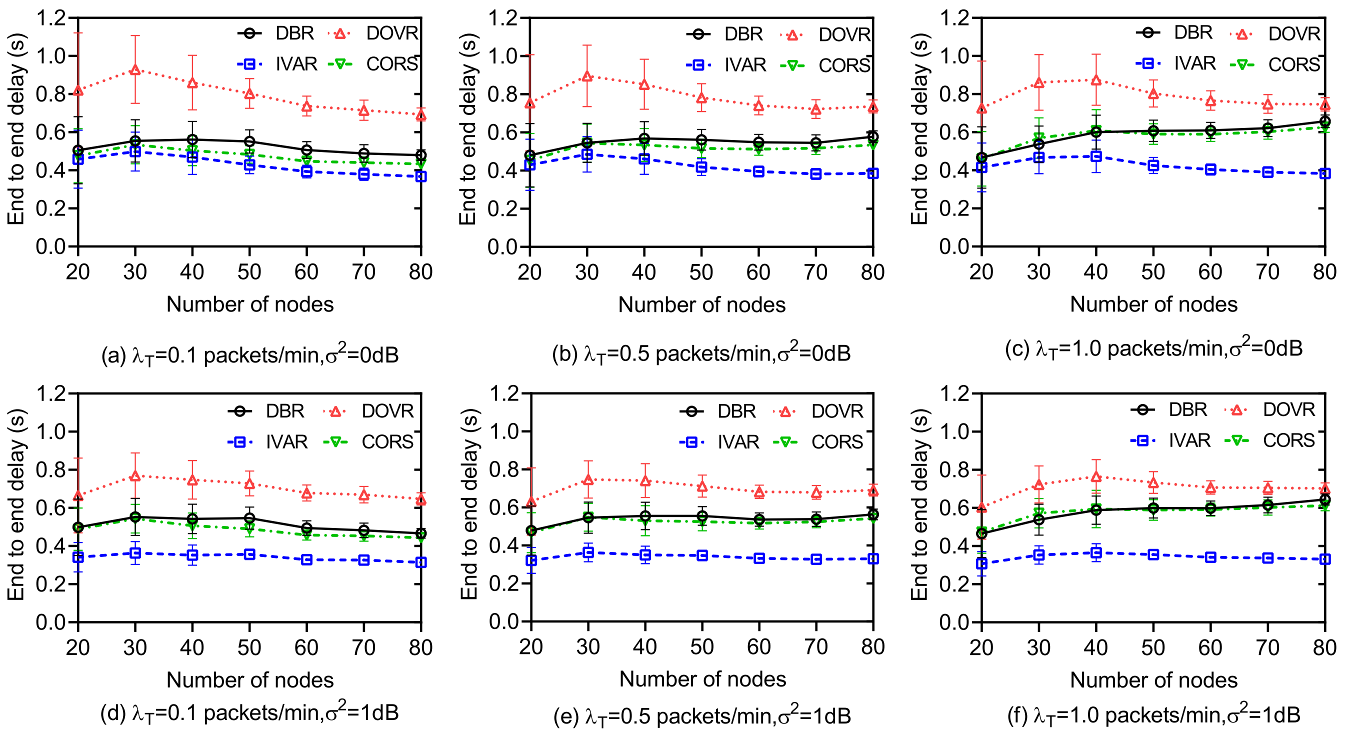


FIGURE 11. Comparisons of latency performances in networks with different densities.

by CORS can significantly decrease the amounts of collisions and interference. Second, opportunistic coding can further improve the delivery performance since there are still some sparse regions due to the asynchronous movement of each node.

As shown in Fig.10, IVAR and DOVR are heavily influenced by channel variations and have worse PDR performances at various network sizes, whereas DBR and CORS still have better delivery performances and can remain stable with different channel variation levels. The reasons for this are as follows: The channel variation decreases the reliability of the candidate sets of the hop count-based protocols. In IVAR, using a strong receiver signal to estimate the distance between each transmitted pair is seriously influenced by channel variation. The strategies used for topology maintenance enhance the robustness of CORS. Thus, CORS is not obviously affected by channel variation. Since the candidate set is not dependent on topological information, DBR can adapt well to channel variation. We can observe from Fig. 10 that in sparse scenarios, the delivery performance of DBR is obviously better than those of IVAR and DOVR but worse than that of CORS when $\sigma^2 = 1\ dB$. This suggests that hop count-based protocols can effectively improve delivery performance by optimizing the topology update mechanism and opportunistic coding in sparse scenarios. As the network density increases, the DBR protocol achieves the best delivery performance. The delivery performance of CORS is close to that of DBR only at low traffic rates since channel variation influences the robustness of CORS at high traffic rates.

As shown in Fig.10, the variance in the PDR performances of the four protocols in sparse scenarios is large and decreases as the network density increases. This suggests that topology variations influence not only hop count-based protocols but also pressure-based protocols. Among these protocols, CORS has the smallest variance in terms of its PDR performances in scenarios with large network densities. This is because the improvement of the candidate set selection process can enhance the robustness of the network performance.

Fig. 11 shows the latency performances of the four protocols for various network sizes at different traffic rates. As shown in Fig. 11, as the network size becomes large, the latency performances of these protocols first increase and then decrease at high network densities. When the network density is lower, the void problem can be significantly improved as the network density increases. Thus, each sink can collect more packets from remote nodes, and this increases the average end-to-end delay. However, as the network density continually increases, the candidate set becomes large such that superior nodes are involved in forwarding. Thus, the average transmission distance or the number of hop counts to each sink is reduced, which, in turn, reduces the average end-to-end delay. We can observe from Fig. 11 that CORS performs better than DOVR but worse than IVAR. First, CORS allows more nodes with larger depths to participate in forwarding, and this increases the average propagation time of a given packet. Second, the backoff strategy introduces more waiting time than other protocols.

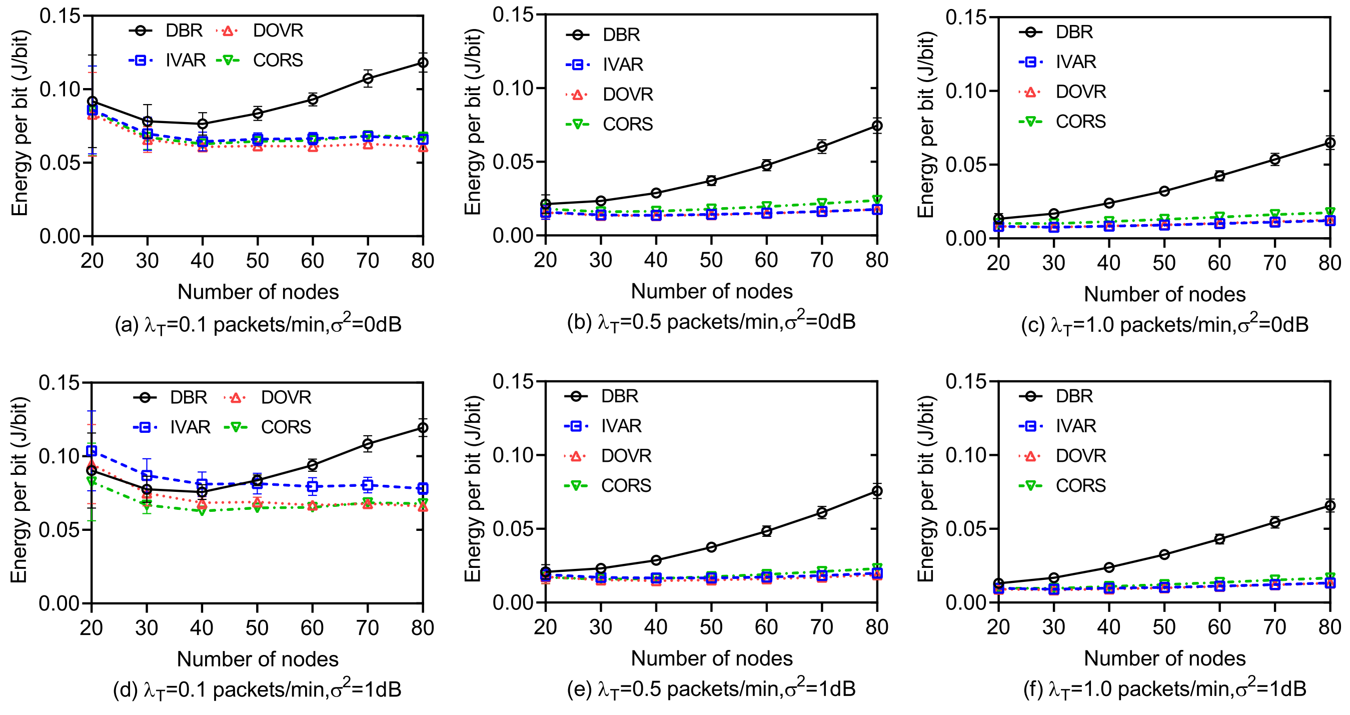


FIGURE 12. **Comparisons of energy efficiency in networks with different densities.**

Third, the decoding algorithm at the base station introduces a decoding delay with regard to some information.

As shown in Fig. 11, IVAR and DOVR perform better in the dynamic channel than in the static channel. In contrast, DBR and CORS perform approximately the same in different channel conditions. This is due to the reasons below. In hop count-based protocols, a remote node beyond the average transmission range can be involved in the candidate set with probability in time-varying channel conditions. Thus, the network latency is decreased when the hop count in the dynamic channel is smaller than that in time-invariant channel conditions. Moreover, as the link worsens, the nodes in subset L within the transmission range may have multihop distances from the current forwarder. Thus, the waiting time for these nodes is significantly smaller than that in time-invariant channel conditions, and this in turn results in a smaller end-to-end delay. However, collisions and multihop interferences can limit the improvement in the delay performance brought by channel variation. In CORS, the topology establishment and maintenance procedures weaken the improvement in network latency. Thus, the delay performance cannot be effectively improved. In DBR, a highly dynamic channel can extend the forward region, and this decreases the forwarding path. However, the hop count to the sink is increased for some packets since some available links fail due to channel variation. These pros and cons ultimately result in slight delay performance changes in different channel conditions.

As shown in Fig. 11, the variance in the latency performances of the four protocols in sparse scenarios is large and decreases as the network density increases. Since the nodes are randomly deployed in each scenario, the distance variance between each transmission pair in sparse scenarios is larger than that in dense scenarios. Thus, the variance of the latency performances in sparse scenarios is significantly greater than that in dense scenarios. We can observe that the latency performances of these three protocols are robust at high network densities, except for that of DOVR.

Fig. 12 shows the energy efficiencies of four protocols for various network sizes at different traffic rates. At low traffic rates, the energy consumption decreases as the network density increases since the void problem is significantly improved. As the network density is continually increased, the energy efficiencies of these protocols change slightly except for that of DBR. This is due to the reasons as below. The traffic rate increases linearly as the network size increases, which results in an exponential increase in the probability of collision. The energy efficiency of DBR is degraded as additional competitions are introduced within smaller time intervals than those of the other three protocols.

At low traffic rates, the energy efficiency of CORS is close to that of DOVR and better than those of IVAR and DBR when $\sigma^2 = 0\ dB$. As shown in Fig. 10, the delivery performance reaches the upper limit as the network size increases to some extent. Thus, the energy efficiency decreases since the coding redundancy significantly increases as the network size increases. At high traffic rates, the energy consumption of CORS is higher than those of DOVR and IVAR because the coding redundancy increases as the probability of coding increases.

At low traffic rates, the energy efficiencies of IVAR and DOVR are degraded in the dynamic channel since the reliability of the candidate set is worsened by channel variation. In contrast, the energy efficiency of DBR is slightly changed as the reliability of the candidate set changes slightly due to channel variation. However, CORS maintains better energy efficiency due to topology maintenance and the flexible coding strategy. The energy efficiencies of the four protocols are highly robust except at low traffic rate. This is because the total transmission bit information is smaller. Thus, a variation may significantly influence the energy spent transmitting 1 bit. In contrast, the energy spent transmitting 1 bit is smaller at higher traffic rates, and this results in slight variations in energy efficiency.

VII. CONCLUSION

In this paper, we proposed an opportunistic routing method called CORS for sparse UWSNs. To alleviate the void problem, we exploited topological information to extend the obtained candidate set. On this basis, a forwarding with opportunistic coding strategy was developed to join interflow network coding and opportunistic routing in CORS. In addition, we developed a sliding window-based coding algorithm to reduce coding overhead and a sliding window-based decoding algorithm to reduce decoding overhead. Simulation results showed that CORS outperforms IVAR, DOVR, and DBR in terms of delivery performance in sparse scenarios. Furthermore, CORS can adapt best to channel variation as compared with IVAR and DOVR. Moreover, CORS has better energy efficiency than the DBR protocol in various scenarios. In our future work, we will introduce channel state estimation to optimize the CORS protocol.

REFERENCES

[1] J. Heidemann, M. Stojanovic, and M. Zorzi, "Underwater sensor networks: Applications, advances and challenges," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 370, no. 1958, pp. 158–175, Jan. 2012.

[2] D. Pompili and I. Akyildiz, "Overview of networking protocols for underwater wireless communications," *IEEE Commun. Mag.*, vol. 47, no. 1, pp. 97–102, Jan. 2009.

[3] A. Caruso, F. Paparella, L. F. M. Vieira, M. Erol, and M. Gerla, "The meandering current mobility model and its impact on underwater mobile sensor networks," in *Proc. IEEE 27th Conf. Comput. Commun. (INFOCOM)*, Apr. 2008, pp. 221–225.

[4] J. Jaffe and C. Schurgers, "Sensor networks of freely drifting autonomous underwater explorers," in *Proc. 1st ACM Int. Workshop Underwater Netw. (WUWNet)*, 2006, pp. 93–96.

[5] C. Jun-Hong, K. Jiejun, M. Gerla, and Z. Shengli, "The challenges of building mobile underwater wireless networks for aquatic applications," *IEEE Netw.*, vol. 20, no. 3, pp. 12–18, May/Jun. 2006.

[6] Y. Noh, U. Lee, S. Han, P. Wang, D. Torres, J. Kim, and M. Gerla, "DOTS: A propagation delay-aware opportunistic MAC protocol for mobile underwater networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 4, pp. 766–782, Apr. 2014.

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

[8] H. Yu, N. Yao, and J. Liu, "An adaptive routing protocol in underwater sparse acoustic sensor networks," *Ad Hoc Netw.*, vol. 34, pp. 121–143, Nov. 2015.

[9] A. Darehshoorzadeh and A. Boukerche, "Underwater sensor networks: A new challenge for opportunistic routing protocols," *IEEE Commun. Mag.*, vol. 53, no. 11, pp. 98–107, Nov. 2015.

[10] R. W. L. Coutinho, A. Boukerche, L. F. M. Vieira, and A. A. F. Loureiro, "Geographic and opportunistic routing for underwater sensor networks," *IEEE Trans. Comput.*, vol. 65, no. 2, pp. 548–561, Feb. 2016.

[11] S. M. Ghoreyshi, A. Shahrabi, and T. Boutaleb, "Void-handling techniques for routing protocols in underwater sensor networks: Survey and challenges," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 800–827, 2nd Quart., 2017.

[12] S. M. Ghoreyshi, A. Shahrabi, and T. Boutaleb, "An inherently void avoidance routing protocol for underwater sensor networks," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2015, pp. 361–365.

[13] Q. Guan, F. Ji, Y. Liu, H. Yu, and W. Chen, "Distance-vector-based opportunistic routing for underwater acoustic sensor networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3831–3839, Apr. 2019.

[14] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft, "XORs in the air: Practical wireless network coding," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 497–510, Jun. 2008.

[15] O. M. Al-Kofahi and A. Kamal, "Network coding-based protection of many-to-one wireless flows," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 5, pp. 797–813, Jun. 2009.

[16] R. Prior, D. E. Lucani, Y. Phulpin, M. Nistor, and J. Barros, "Network coding protocols for smart grid communications," *IEEE Trans. Smart Grid*, vol. 5, no. 3, pp. 1523–1531, May 2014.

[17] Y. Yan, B. Zhang, J. Zheng, and J. Ma, "CORE: A coding-aware opportunistic routing mechanism for wireless mesh networks [accepted from open call]," *IEEE Wireless Commun.*, vol. 17, no. 3, pp. 96–103, Jun. 2010.

[18] M. K. Han, A. Bhartia, L. Qiu, and E. Rozner, "O3: Optimized overlay-based opportunistic routing," in *Proc. 12th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2011, pp. 1–11.

[19] K. Chung, Y.-C. Chou, and W. Liao, "CAOR: Coding-aware opportunistic routing in wireless ad hoc networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 136–140.

[20] H. Liu, H. Yang, Y. Wang, B. Wang, and Y. Gu, "CAR: Coding-aware opportunistic routing for unicast traffic in wireless mesh networks," *J. Netw. Syst. Manage.*, vol. 23, no. 4, pp. 1104–1124, Oct. 2015.

[21] P. Xie, J.-H. Cui, and L. Lao, "VBF: Vector-based forwarding protocol for underwater sensor networks," in *Proc. Int. Conf. Res. Netw.* Berlin, Germany: Springer, 2006, pp. 1216–1221.

[22] N. Nicolaou, A. See, P. Xie, J.-H. Cui, and D. Maggiorini, "Improving the robustness of location-based routing for underwater sensor networks," in *Proc. OCEANS-Eur.*, Jun. 2007, pp. 1–6.

[23] Q. Wang, J. Li, Q. Qi, P. Zhou, and D. O. Wu, "A game-theoretic routing protocol for 3-D underwater acoustic sensor networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9846–9857, Oct. 2020.

[24] H. Yan, Z. J. Shi, and J.-H. Cui, "DBR: Depth-based routing for underwater sensor networks," in *Proc. Int. Conf. Res. Netw.* Berlin, Germany: Springer, 2008, pp. 72–86.

[25] A. Wahid, S. Lee, H.-J. Jeong, and D. Kim, *EEDBR: Energy-Efficient Depth-Based Routing Protocol for Underwater Wireless Sensor Networks*. Berlin, Germany: Springer, 2011, pp. 223–234.

[26] H. Yu, N. Yao, T. Wang, G. Li, Z. Gao, and G. Tan, "WDFAD-DBR: Weighting depth and forwarding area division DBR routing protocol for UASNs," *Ad Hoc Netw.*, vol. 37, pp. 256–282, Feb. 2016.

[27] Y. Noh, U. Lee, S. Lee, P. Wang, L. F. M. Vieira, J.-H. Cui, M. Gerla, and K. Kim, "HydroCast: Pressure routing for underwater sensor networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 1, pp. 333–347, Jan. 2016.

[28] M. Ayaz and A. Abdullah, "Hop-by-hop dynamic addressing based (H2-DAB) routing protocol for underwater wireless sensor networks," in *Proc. Int. Conf. Inf. Multimedia Technol.*, 2009, pp. 436–441.

[29] M. Ayaz, A. Abdullah, and I. Faye, "Hop-by-hop reliable data deliveries for underwater wireless sensor networks," in *Proc. Int. Conf. Broadband, Wireless Comput., Commun. Appl.*, Nov. 2010, pp. 363–368.

[30] Y. Noh, U. Lee, P. Wang, B. S. C. Choi, and M. Gerla, "VAPR: Void-aware pressure routing for underwater sensor networks," *IEEE Trans. Mobile Comput.*, vol. 12, no. 5, pp. 895–908, May 2013.

[31] M. Stojanovic, "On the relationship between capacity and distance in an underwater acoustic communication channel," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 11, no. 4, pp. 34–43, Oct. 2007.

[32] L. M. Brekhovskikh, Y. P. Lysanov, and R. T. Beyer, *Fundamentals of Ocean Acoustics*. Berlin, Germany: Springer, 1991.

[33] C. Tapparello, P. Casari, G. Toso, I. Calabrese, R. Otnes, P. van Walree, M. Goetz, I. Nissen, and M. Zorzi, "Performance evaluation of forwarding protocols for the RACUN network," in *Proc. 8th ACM Int. Conf. Underwater Netw. Syst. (WUWNet)*, 2013, pp. 1–8.

[34] J. Partan, J. Kurose, B. N. Levine, and J. Preisig, "Low spreading loss in underwater acoustic networks reduces RTS/CTS effectiveness," in *Proc. 6th ACM Int. Workshop Underwater Netw. (WUWNet)*, 2011, pp. 1–8.

[35] D. J. C. MacKay, "Fountain codes," *IEE Proc. Commun.*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[36] O. Trullols-Cruces, J. M. Barcelo-Ordinas, and M. Fiore, "Exact decoding probability under random linear network coding," *IEEE Commun. Lett.*, vol. 15, no. 1, pp. 67–69, Jan. 2011.

[37] P. Qarabaqi and M. Stojanovic, "Statistical characterization and computationally efficient modeling of a class of underwater acoustic communication channels," *IEEE J. Ocean. Eng.*, vol. 38, no. 4, pp. 701–717, Oct. 2013.

[38] M. Z. Zamalloa, K. Seada, B. Krishnamachari, and A. Helmy, "Efficient geographic routing over lossy links in wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 4, no. 3, pp. 1–33, May 2008.

[39] *Evologics S2C M HS Modem.* Accessed: Dec. 2020. [Online]. Available: https://evologics.de/acoustic-modem/hs

DANFENG ZHAO received the B.S. and Ph.D. degrees from the College of Information and Communication Engineering, Harbin Engineering University, Harbin, China, in 1983 and 2002, respectively. He had been a Professor and an Assistant Dean of the College of Information and Communication Engineering, Harbin Engineering University, in 2002 and from 2002 to 2012, respectively. From 2009 to 2012, he was the Director of the Teaching Demonstration Center of National Electrical and Electronic Experimental, Harbin Engineering University. In 2010, he was an Adjunct Leader of the Course for Electrical and Electronic Engineering, National Teaching Team Foundation Course. His current research interests include modern communication systems, communications and signal processing, wireless communications systems, radio and acoustic sensor networking, high performance coding and modulation, and video signal processing.

GUIYANG LUN received the B.S. and M.S. degrees in electronic and information engineering from Harbin Engineering University, Harbin, Heilongjiang, China, in 2011 and 2013, respectively, where he is currently pursuing the Ph.D. degree under the guidance of Dr. D. Zhao. His current research interests include underwater wireless sensor networks, opportunistic routing, and network coding.

RUI XUE received the M.E. and Ph.D. degrees from Harbin Engineering University, Harbin, China, in 2006 and 2009, respectively. From July 2011 to July 2012, he was a Visiting Scholar with the Nonlinear Signal Processing Laboratory, The University of Melbourne, Australia. From August 2010 to September 2013, he was a Postdoctoral Fellow with the College of Automation, Harbin Engineering University, for satellite navigation research. His research interests include radio mobile communication systems, satellite communication systems and satellite navigation and positioning, and include error-correcting codes, high spectral efficiency modulation, coded modulation, iterative decoding and detection, and so on.