Isufi *et al. EURASIP Journal on Advances in Signal Processing* (2016) 2016:52
DOI 10.1186/s13634-016-0346-y

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Advanced flooding-based routing protocols for underwater sensor networks

Elvin Isufi[1]*, Henry Dol[2] and Geert Leus[1]

Abstract

Flooding-based protocols are a reliable solution to deliver packets in underwater sensor networks. However, these protocols potentially involve all the nodes in the forwarding process. Thus, the performance and energy efficiency are not optimal. In this work, we propose some advances of a flooding-based protocol with the goal to improve the performance and the energy efficiency. The first idea considers the node position information in order to reduce the number of relays that may apply flooding. Second, a network coding-based protocol is proposed in order to make a better use of the duplicates. With network coding, each node in the network recombines a certain number of packets into one or more output packets. This may give good results in flooding-based protocols considering the high amount of packets that are flooded in the network. Finally, a fusion of both ideas is considered in order to exploit the benefits of both of them.

Keywords: Underwater communications, Flooding-based routing, Network coding, Geographical routing, Implicit acknowledgement

1 Introduction

Starting from the first underwater telephone, developed by the Naval Underwater Sound Laboratory (USA) [1], many research efforts have been put in underwater communications for both civil and military applications [2]. Acoustic technology is mostly preferred for communication distances that exceed about a hundred meters. However, in contrast to terrestrial radio-frequency communications, underwater acoustic links are characterized by long propagation delays, low data rates, limited and variable bandwidth, and high bit error rates [3, 4]. According to [5], the attenuation of an underwater acoustic link increases exponentially with the distance, and in [6], it has been shown that we also pay in terms of bandwidth for greater transmission ranges. In several underwater communication scenarios, we may be interested in reaching distances longer than the range of a direct transmission link, or large areas need to be covered. In these cases, an extension to an underwater acoustic communication *network* is required (i.e., multihop transmission instead of a single direct transmission), which brings benefits in terms of energy and capacity.

On the other hand, underwater acoustic networks bring new issues, and thus efficient routing protocols are required to determine the path that the packets must follow to reach the destination. We redirect the reader to [3, 7] and references therein for a deeper analysis and overview about underwater routing protocols and to [8] for design guidelines about opportunistic routing. Irrespective of the routing protocol, or the application scenario, one of the main goals is to obtain a high packet delivery ratio (PDR) while keeping the end-to-end delay and energy consumption limited. One class of routing protocols, which can be used in underwater scenarios, consists of flooding-based protocols. These protocols may be preferred in networks where the nodes are generally not static, and when the communication links face outages, meaning that continuously updating the routing table may reduce the overall throughput. The performance of flooding-based protocols starts degrading when the network becomes overloaded by traffic. Thus, in order to avoid this situation, and to reduce the number of collisions, the number of duplicates that are flooded in the network must be kept limited.

*Correspondence: e.isufi-1@tudelft.nl
[1] Faculty of Electical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD, Delft, The Netherlands
Full list of author information is available at the end of the article

In [9], a duplicate reduction flooding-based protocol (called *Dflood*) has been proposed[1]. Here, the nodes use some backoff time for each information packet, which is continuously adapted when a duplicate is overheard. Even considering these duplicate reduction policies applied by Dflood, it may happen that in certain scenarios, the number of packets forwarded by the network, i.e., *the energy consumption* is still high. For more details about the Dflood protocol, we redirect the reader to [9] and [10], where more information is provided for all protocol layers. In this work, we will mostly focus on advancing the performance (improve the PDR), and the energy efficiency of Dflood.

The first idea is to reduce *the number of duplicates* that are forwarded by the network, using node position information. Our motivation comes from the fact that Dflood does not distinguish between a source node that is closer to the destination and another one that might be further. As a result, whatever the physical position of the source, the entire network is prospected to take part in the relaying process. In large networks, and when the sources are close to the destination, the resulting amount of packets forwarded by the network may be unacceptable. To reduce the energy consumption, we propose an enhancement of Dflood exploiting the location information of the node that is transmitting and the final destination position. This information has been used in other approaches such as [11] and [12]. The main difference with the last approach [12] is that our nodes are equipped with omnidirectional hydrophones, so all the neighbours within the transmission range can receive the transmitted packet. The way in which we use location information is the main difference with the first approach [11]. Instead of creating a straight pipe from the source node to the destination, and to allow only the nodes inside the pipe to take part in the forwarding, we consider the possibility to involve also other nodes when necessary. More details will be provided in the next section. Furthermore, we have extended our approach with the use of an implicit acknowledgement (ACK). This is an efficient retransmission strategy, where the ACK is not explicitly sent by the receiver node. More specifically, if the sender node, within a certain interval of time, does not overhear the transmission of that specific packet from one of its neighbours (remember that the nodes are equipped with omnidirectional hydrophones), it will retransmit another copy. This strategy helps improving the PDR, as well as reducing the end-to-end delay and energy consumption with respect to the other automatic-repeat-request (ARQ) schemes.

Our second contribution is motivated by the fact that the number of replicas that are flooded in the network and received by the destination is still high. Instead of trying to reduce them further, we have considered the possibility that these packets may share the information of more than one packet. Basically speaking, we propose network coding (NC) [13], in order to have more information flooding in the network than replicas. With NC, each node in the network is allowed to recombine a certain number of packets into one or more output packets, instead of simply forwarding each of them. It is clear that this increases the robustness of the transmission, especially in a flooding fashion. *Linear* NC has been proposed recently, where the output packets are linear combinations of the packets presented in the node's buffer. It has also been used in underwater sensor networks to potentially increase the PDR, the energy efficiency, and the throughput, as well as to reduce the end-to-end delay [14–16]. It has shown promising results in both simulations [17], and real experimental trials [18, 19]. This motivated us to consider NC also in flooding-based protocols, where many packets flood in the network and the sharing of information between them will potentially improve the information at the destination. However, we should be aware of the fact that being inspired by a flooding-based approach, the number of replicas must be kept limited in order to be energy-efficient. For this reason, we propose to fuse linear NC with Dflood in order to gain the benefits from both approaches, i.e., more robustness from the NC and a reduced energy consumption using the Dflood idea. Our idea does not simply consist of using NC on top of Dflood. Instead, we have reviewed the latter's rules and changed them in order to exploit the potential of NC. This is the main difference with the approach in [14]. The key differences with [15] and [16] are, respectively, that our approach is developed for any scenario with a sink node (not only for a chain topology), and that the packets arrive at the next relay node with a different and variable delay. Next, we have considered such an approach in conjunction with geographical information, as discussed earlier. For the considered scenario, simulation results show an increment of up to 10 % of the PDR, with respect to the original Dflood, for low packet error rates (PERs) and a low traffic load. At the same time, the energy efficiency of the network is increased.

The rest of this paper is organized as follows. In the next section, we will present our geographical protocol; the NC-based solution will be explained in Section 3; in Section 4, we will show the simulation results; and the work will be concluded in Section 5.

2 Geographical Dflood

2.1 Proposed protocol

Geographical Dflood (GDflood) uses the node position information in order to reduce the number of relays that take part in the forwarding process. With GDflood, the transmitting node makes use only of its own position and the final destination position. Using this information during the relaying, it is more likely that only those nodes

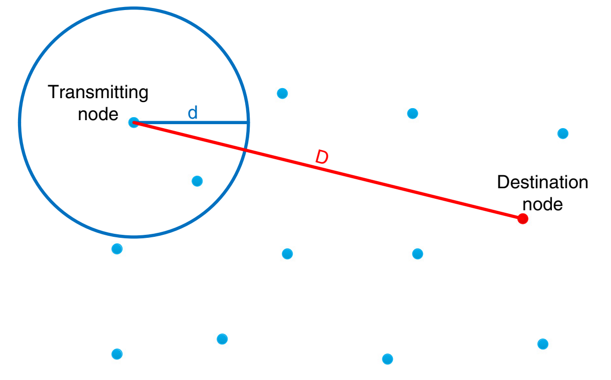


Fig. 1 Illustration of the GDflood parameters

that are closer to the destination take part in the forwarding process. GDflood uses the same network header, as Dflood, composed by *a source address*, *a destination address*, *a sequence number*, and *a hop count*. To distinguish between different packets, the first three fields form a *unique ID* in the network.

We assume that in a single transmission (i.e., 1 hop), a node can send a packet to its neighbors within a distance d. When a node has a packet to send, it first calculates the distance D, between itself and the destination, as illustrated in Fig. 1. Then, this distance is quantized into hop counts

$$D_{\mathrm{HC}} = \lceil D/d \rceil \,, \tag{1}$$

where $\lceil \cdot \rceil$ indicates the ceiling operator. The transmitting node puts the value $D_{\mathrm{HC}} + f_{\mathrm{n}}$ in the hop count field, where $f_{\mathrm{n}} \leq 0$ is a common term for all nodes that influences the number of relays that may consider the packet for the next transmission. The role of this parameter is to keep the number of relays limited (see later on). Only if the node is the source node, it will consider a redundancy factor $f_{\mathrm{s}} \geq 0$, for its own packets. The hop count of these packets will be $D_{\mathrm{HC}} + f_{\mathrm{n}} + f_{\mathrm{s}}$ (see also later on). After setting the hop count, the node sends the packet immediately to the MAC layer. Depending on the MAC protocol used, if the packet is not directly transmitted but sent back to the upper layers, the above mechanism has to be repeated. This is important because the moving nodes change continuously their position, and as a consequence D_{HC} changes. GDflood is explained in detail below in relation to the role that a specific node has in the transmission, i.e., if it is *the source* of the packet, *the destination* node, or *a relay* node, see also Fig. 2.

Source node: In case the node is the source node, the packets to be transmitted are received from the upper layers. The source will schedule the packet directly, so the backoff time, T_{back}, in this case is the next available time instant. When this time is reached, the source node calculates the D_{HC}, as explained before enters as hop count $D_{\mathrm{HC}}+f_{\mathrm{n}}+f_{\mathrm{s}}$, and sends the packet down to the MAC layer.

Destination node: In case a node receives a packet intended for itself, i.e., if it is the destination node, it will immediately broadcast a receive notification (RN). The nodes that receive the RN will stop relaying the packet with the same unique ID. The RN packet will not be forwarded.

Relay node: When the node is a relay, it will first check if the packet has been received before or not. In case the packet is a duplicate, and already transmitted, it will not be forwarded anymore.

In case the packet is a duplicate but scheduled to be transmitted, the node will apply some duplicate policies. First, it will check if it has to stop taking part in the relaying process. This is done by uniformly drawing a random number $\rho \in [0,1]$ and by checking if

$$n_{\mathrm{d}} > N_{\mathrm{dupl}} - \rho, \tag{2}$$

where n_{d} is the number of duplicates collected till that moment, and N_{dupl} is defined as the maximum number of duplicates (in practice, it can be defined also as a rational number). If the test fails, the relay node will add a T_{dupl} to the scheduled time for that packet.

In case the packet is not a duplicate, the relay node will check if it can take part in the forwarding. This is done by checking if $D_{\mathrm{HC}} \leq$ hop count, where D_{HC} is the distance in hop counts of the relay node to the destination. In case the node is allowed to take part in the relaying process, the packet will be scheduled to be transmitted after a random time T_{back}, uniformly drawn in $[T_{\mathrm{min}}, T_{\mathrm{max}}]$. The new packet will have $D_{\mathrm{HC}} + f_{\mathrm{n}}$ as new hop count value.

Considering the behaviour of the relay nodes, it should be clear that a positive value of f_{n} will increase the hop count of the newly forwarded packet with respect to the received one. In this way, at each forwarding stage, more nodes will get involved and this will cause flooding in the whole network. As a result, we always consider a negative f_{n}. In contrast, the parameter f_{s} is useful to be positive, especially in those scenarios where the source node is characterized by low connectivity. In other words, in the first hop transmission, the source node might also need those neighbours that are not closer to the destination than itself. In this way, more relays will participate in the first transmissions, and their number will reduce while we get closer to the destination in the next stages.

2.2 Implicit ACK

Considering the fact that the underwater sensors are mostly equipped with omnidirectional hydrophones, and that the information is broadcast to the nodes that are within the transmission range, an implicit ACK strategy

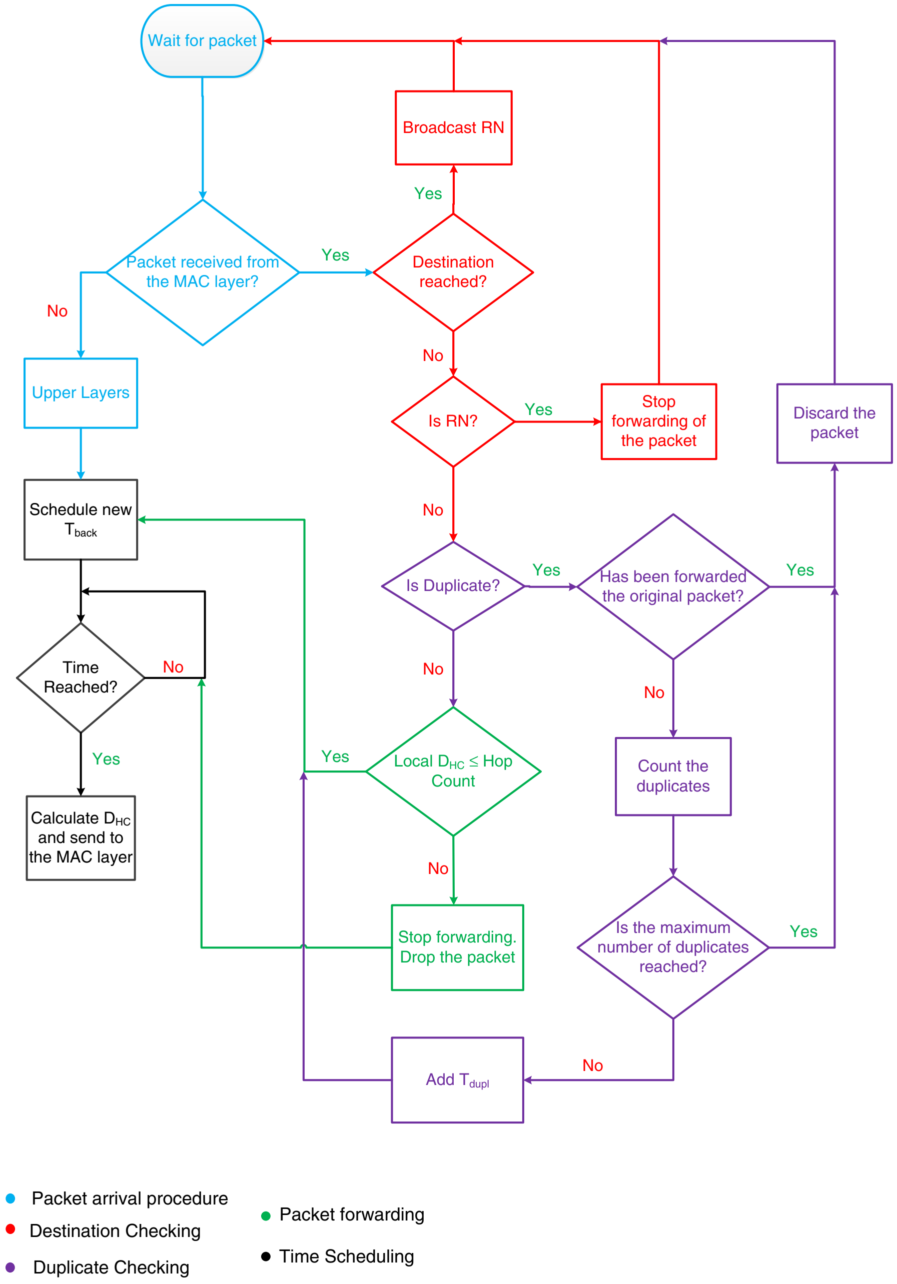


Fig. 2 GDflood flowchart

can be easily adopted. This way of re-transmission can be used by the nodes to detect if a packet has been successfully received by one of its neighbours. In this manner, we use a link-by-link ACK without the need of transmitting special packets. If, after some time from the moment of transmitting the packet, the node does not overhear any forwarding by one of its neighbors, it will retransmit another copy. This procedure can be repeated by all the nodes till the packet reaches the destination. To be energy-efficient, the number of retransmissions must be kept limited. For each packet forwarded, each node in the network starts a local timer, called the ACK time. If during this time, the node under consideration overhears a transmission of that packet, by any of its neighbors, it will not retransmit other copies. Otherwise, at the end of the ACK time, the node will retransmit another copy and will start another ACK time. This procedure will be done for a fixed number of re-transmissions. In [10], a random selection of the implicit ACK time has been proposed for the Dflood protocol. We will use the same approach in order to have a direct comparison between the two protocols. The ACK time is selected as

$$\tau = \text{rand}\left(\left[\frac{t_{\text{delay}}}{2},\ t_{\text{delay}}\right]\right), \tag{3}$$

where τ is the ACK time, t_{delay} is the delay time relative to the retransmissions (retx.), and rand $[a, b]$ selects a random number, uniformly drawn in $[a, b]$.

3 Network coding Dflood

In this section, we will explain our network coding (NC) approach. With NC, the nodes encode the incoming packets into one or more output packets instead of using the classic store and forward approach. In this way, the original information is shared among the encoded packets, and thus the destination is more likely to receive a piece of information instead of replicas of the same packet. We will use *linear* NC due to its effectiveness and applicability in underwater sensor networks. With linear NC, the output packets are a linear combination of the packets present in the node's buffer. Together with the Dflood idea, we aim to keep the number of packets that are flooded in the network limited, yet improving the PDR.

3.1 Linear network coding background

3.1.1 Encoding

Let us consider a network where a source node has a set of g packets to send $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_g]^T$, where $\mathbf{x}_i \in \{0,1\}^{L_b \times 1}$, $(\cdot)^T$ indicates the transpose operator and L_b is the number of bits that each packet is composed of. We interpret s consecutive bits as a symbol over the field $\mathbb{F}_{2^s}$, so each packet consists of L_b/s symbols. In linear NC, the source node linearly combines the g original packets into h encoded packets $\mathbf{Y}_1 = [\mathbf{y}_{11}, \ldots, \mathbf{y}_{1h}]^T$, with $h \geq g$. The operations are carried out over the field $\mathbb{F}_{2^s}$ [20]. For each encoded packet $\mathbf{y}_{1i}$, the node selects the respective encoding vector $\mathbf{e}_{1i}$ over the field $\mathbb{F}_{2^s}$, [21], and the output encoded packets are obtained as

$$\mathbf{Y}_1 = \mathbf{E}_1\mathbf{X}, \tag{4}$$

where $\mathbf{E}_1$ $(h \times g)$ is the matrix that contains as the ith row the encoding vector $\mathbf{e}_{1i}^T$. Note that both $\mathbf{Y}_1$ and $\mathbf{E}_1$ will be transmitted. With NC, not only the source, but also the relay nodes can re-encode the received packets. Let us consider a relay node that receives a set of encoded packets $(\mathbf{e}_{11}, \mathbf{y}_{11}), (\mathbf{e}_{12}, \mathbf{y}_{12}), \ldots, (\mathbf{e}_{1k}, \mathbf{y}_{1k})$, with $k \leq h$. The relay node will consider the received packets $\tilde{\mathbf{Y}}_1 = [\mathbf{y}_{11}, \ldots, \mathbf{y}_{1k}]^T$ and the corresponding encoding matrix $\tilde{\mathbf{E}}_1 = [\mathbf{e}_{11}, \ldots, \mathbf{e}_{1k}]^T$ to re-encode these packets into l output packets, $\mathbf{Y}_2 = [\mathbf{y}_{21}, \ldots, \mathbf{y}_{2l}]^T$, with $l \leq k$. It is clear that $\tilde{\mathbf{Y}}_1$ and $\tilde{\mathbf{E}}_1$ are, respectively, submatrices composed of k rows, of $\mathbf{Y}_1$ and $\mathbf{E}_1$. For each output packet $\mathbf{y}_{2i}$, the relay node selects locally a set of coefficients $\mathbf{e}_{2i}^{'}$, in $\mathbb{F}_{2^s}$, and the output packets are computed as

$$\mathbf{Y}_2 = \mathbf{E}_2'\tilde{\mathbf{Y}}_1, \tag{5}$$

where $\mathbf{E}_2'$ is the matrix that contains as the ith row the local re-encoding vector $\mathbf{e}_{2i}'^T$. Substituting the relationship of $\tilde{\mathbf{Y}}_1$ with the original information $\mathbf{X}$ in (5), we obtain a direct relationship of the new information vectors with the original packets

$$\mathbf{Y}_2 = \mathbf{E}_2'\tilde{\mathbf{E}}_1\mathbf{X} = \mathbf{E}_2\mathbf{X}. \tag{6}$$

The generic ith output packet $\mathbf{y}_{2i}$ and the corresponding encoding vector $\mathbf{e}_{2i}^T$, which will be transmitted, are obtained as

$$\mathbf{y}_{2i}^T = \mathbf{e}_{2i}'^T\tilde{\mathbf{E}}_1\mathbf{X} = \mathbf{e}_{2i}^T\mathbf{X} \tag{7}$$

This procedure can be repeated by all the nodes in the network until the packets reach the destination.

3.1.2 Decoding

In order to decode the data, the destination needs a sufficient number of packets. Let us assume for simplicity that the sink receives a set of m packets after passing n stages $(\mathbf{e}_{n1}, \mathbf{y}_{n1}), (\mathbf{e}_{n2}, \mathbf{y}_{n2}), \ldots, (\mathbf{e}_{nm}, \mathbf{y}_{nm})$. In order to decode these packets and retrieve the original information, i.e., $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_g]^T$, the node must solve the linear system

$$\tilde{\mathbf{Y}}_n = \tilde{\mathbf{E}}_n\mathbf{X}, \tag{8}$$

where $\tilde{\mathbf{Y}}_n = [\mathbf{y}_{n1}, \ldots, \mathbf{y}_{nm}]^T$ contains the received packets and $\tilde{\mathbf{E}}_n = [\mathbf{e}_{n1}, \ldots, \mathbf{e}_{nm}]^T$ contains the encoding vectors. This linear system can be solved only if the destination collects enough packet, i.e., $n \geq g$, and at least g combinations must be independent. This means that the rank of the matrix $\tilde{\mathbf{E}}_n$ must be g. Considering the computational cost, the inversion of a matrix is related to the cubic power of its rank. Thus, the source groups the packets in so-called generations, and the encoding process is limited only to the packets of the same generation [21].

Another key element for decoding the information are the encoding coefficients. In [22], a randomized selection of the encoding coefficients has been proposed. Each node in the network selects randomly the coefficients, uniformly distributed in $\mathbb{F}_{2^s}$. This is a liked strategy since it helps to have an independent and decentralized network. The benefits are enhanced in underwater sensor networks where transmitting the information from a central unit may take too long. However, selecting randomly the coefficients may lead to linearly dependent combinations, which happens with a probability related to the field size s, [22]. However, [21] has shown that in practice, $s = 8$ is sufficient to have a full rank decoding matrix with very high probability. Considering that the coefficients are chosen locally at each node, the encoding vectors must be included in the packet headers. Obviously, this brings an increment to the overhead, which grows linearly with the generation size. This is because the higher the generation size, the longer the dimension of each encoding vector in order to contain all the entries selected for encoding the information packets into one specific output packet.

3.2 Proposed protocol

Also here, we have considered the same network header as Dflood, with the note that the sequence number is now called generation number. Thus, the *unique* ID refers to the packets of the same generation. So, more packets will hold the same ID. However, this does not give any problems, since we are interested in receiving g independent packets, instead of receiving each of them independently. After decoding, the packets can be sorted in the same order as produced by the application layer of the transmitter. This can be done using the information of the upper layers, or by splitting the b bits of the generation number in two sub-fields. The first one with

$$b_1 = \lceil \log_2 g \rceil \tag{9}$$

bits will be used to sort the packets inside the generation, and the second with

$$b_2 = b - b_1 \tag{10}$$

bits are used for the effective generation number.

We have used the same terminology as in previous scientific works about NC. The *innovative packets* are those which are not a linear combination of the packets present in the node buffer[2]. On the other hand, *non-innovative packets* are those packets that can be expressed as a linear combination of the packets that the node keeps. In order to exploit the benefits of NC, we have changed the Dflood rules, and adapted them into a generation fashion. In this way, we aim to reduce the number of non-innovative packets. On top of that, we want to transmit the necessary amount of innovative packets considering that we are working in a flooding fashion. In the following, we will explain how each node acts in the network, mainly focusing on the novelty of this approach.

Source node: The source node will start transmitting when it receives g packets from the upper layers. It will encode them into h output packets and send them down to the MAC layer with hop count equal to 1. In this way, the source performs NC to the desired group of packets. There might also be cases when there are less than g packets to transmit. In this case, the upper layers may inform the encoder to process a smaller group of packets. This will save time and keep the end-to-end delay limited. To contrast harsh environments, it is preferable to have $h > g$. This is because the first hop is the bottleneck of the overall transmission. If all the source's 1-hop neighbours do not receive g independent packets, the overall transmission will result in a waste of energy since the destination cannot retrieve the original information.

Destination node: The destination, on the other side, will not broadcast an RN for each packet received. Instead, it will inform its neighbours when g-independent packets are received and the original information is recovered.

Relay node: The relay node's behaviour is more complicated. Each time it receives a packet of a generation for the first time, i.e., an innovative packet, it will schedule it for forwarding after the relative backoff time, still uniformly drawn in $[T_{\min}, T_{\max}]$. If during the waiting time, the relay receives an innovative packet with the same ID, it will encode this packet together with the packet in the buffer into two new ones. One of them will replace the packet scheduled to be transmitted, and meanwhile the other will be scheduled after the relative backoff time. This process continues whenever an innovative packet with the same ID arrives. So if a new innovative packet arrives, it will be re-encoded together with all the packets with the same ID in the buffer. The number of new re-encoded packets is equal to the number of packets in the buffer not yet transmitted (these will replace the packets in the buffer not yet transmitted) plus one (this packet will receive a new relative backoff time). Note that, in this approach, we assume that even when a packet is transmitted, it will not be removed from the buffer and it will be used for re-encoding packets of the same ID that arrive later.

It may happen that the backoff time expiries without receiving an innovative packet. In order to keep the end-to-end delay limited, and to allow also more nodes to take part in the forwarding, the packet will be transmitted by the relay node without extra encoding.

In this approach, we have considered also a maximum number of times that a packet will be encoded in the network, $H_{\max}$. So the relay will create a new coded packet, if the received one is innovative and with hop count lower than or equal to $H_{\max}$. As before, all the packets in the buffer will be encoded together with the new one received. In case the hop count is higher than $H_{\max}$, the received innovative packet will be considered only to update the scheduled packets and a new one will not be created. In case a new packet is created, the hop count of this packet will be increased by one with respect to the packet received.

The *relay node* will apply the duplicate policy when non-innovative packets are received. First of all, when it has forwarded g packets with the same ID, it will drop any non-innovative packet received. In case not all the packets of a generation are forwarded, and a non-innovative packet is received, the forwarding will be delayed by T_{dupl}. In the NC-Dflood case, there are many ways to do this. We propose two: (1) delay by T_{dupl} only the first packet to be sent, with the same ID and (2) delay by T_{dupl} all the packets scheduled to be sent with the same ID. Depending on the specific scenario, one should simulate both cases and choose the solution that gives the best performance. We refer the reader to [23] for more details on how the selection is done for this particular case.

As before, the number of non-innovative packets will be counted in n_{d}. The relay will try for each received non-innovative packet to quit the forwarding for that generation, if

$$n_{\mathrm{d}} > N_{\mathrm{dupl}} - \rho. \tag{11}$$

In this case, ρ is drawn uniformly in $[0, R]$, and for N_{dupl}, we have proposed two approaches. In the first one, N_{dupl} is related by a constant c to the generation size,

$$N_{\mathrm{dupl}} = c \cdot g, \tag{12}$$

and $R = g$; in the second approach, it is related to the number of independent packets that the relay has collected till that moment,

$$N_{\mathrm{dupl}} = c \cdot \mathrm{rank}(\tilde{\mathbf{E}}) \tag{13}$$

and $R = \mathrm{rank}(\tilde{\mathbf{E}})$.

Finally, we will extend this approach with position information, as in GDflood, in order to reduce even more the energy consumption in the network. In this case, the hop count of the packets is substituted by the quantized distance to the destination, together with the factors f_{n} and f_{s}. In the latter case, each node in the network that receives a packet will apply the duplicate reduction policy if it is a non-innovative packet, or will use it for re-encoding the packets present in the buffer if it is an innovative packet. But, a new encoded packet will be created if the D_{HC} local of this node is lower than or equal to the hop count value. Now it is more clear why the parameter f_{n} can generally not have a positive value. If all the nodes put $f_{\mathrm{n}} > 0$, then an avalanche effect will be created and all the nodes will put a hop count value larger than the actual hop count contained in the packet. So, the whole network will be involved. The parameter $f_{\mathrm{s}} > 0$ can be useful in those cases when the source node has low connectivity. In the NC case, this is even more useful to contrast also the bottleneck effect that is created in the first hop transmission.

4 Simulations

4.1 Simulation setup

4.1.1 *Considered scenario and traffic model*

To compare the protocol performance, we have considered the scenario illustrated in Fig. 3 composed of 22 sea bottom nodes and 1 autonomous underwater vehicle (AUV). For simplicity, we have considered a regular network grid with an intra-node distance of 3 km. The red line is the trajectory of the AUV, which makes a round trip from checkpoint A to B and back with a speed of 4 knots. We have considered three source nodes, node 1, 10, and the AUV, and one destination, node 22. We have compared our solutions with the Dflood protocol. For the Dflood and GDflood, we have assumed the same traffic rate at the application layer. The arrival times of the packets are generated from a Poisson process with the λ parameter defined as

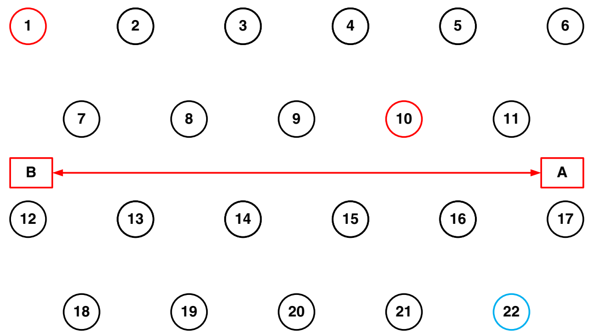


Fig. 3 Network topology. The nodes are deployed in a regular grid, with an intra-node distance of 3 km in a area of 20.25 × 15 km. The network is composed of 22 sea bottom nodes and one moving node

$$\lambda = L_{\mathrm{b}}/r, \tag{14}$$

where L_{b} is assumed to be 160 bits and r (in bits per second) is a simulation parameter that influences the traffic introduced in the network by the application layer of the source nodes. If NC is applied, λ is assumed to be

$$\lambda = (L_{\mathrm{b}} \cdot g)/r \tag{15}$$

since, in this case, it is considered that the application layer produces g packets and sends them down to the MAC layer. The assumption to make the arrival times between sets g times bigger, is done to ensure the same traffic for both cases. However, we have to note that the NC simulations will be affected by more interferences considering that the packets are sent in a shorter time interval.

4.1.2 *Physical and MAC layer*

For our simulations, we have assumed a PER $= p$, common to all the links present in the network. In this way, the evaluated performance can be for different physical transmission schemes. The transmission data rate of the physical layer is considered in such a way that the packet duration is one second. This assumption is considered for simplicity, but nevertheless, it is justified in underwater communications, since the packet lengths are very short and the bit rates are low [24], e.g., in [9], a packet length of 160 bits and a bit rate of 200 bits/s is considered for Dflood. The transmission power is assumed to offer a PER $= p$ for links up to 3 km. Farther nodes receive a packet erroneously with probability one.

In the MAC layer, a simple unslotted ALOHA protocol is considered. We did not consider any carrier sensing

Table 1 Protocol parameters used in the simulations. N/A means that the parameter is not applicable for that protocol

Protocol	$T_{\min}$	$T_{\max}$	T_{dupl}	N_{dupl}	$H_{\max}$	f_{n}	f_{s}	No. of retx	t_{delay}
Dflood	0 s	50 s	35 s	2.5	N/A	N/A	N/A	N/A	N/A
GDflood	0 s	50 s	35 s	2.5	N/A	0	2	N/A	N/A
Dflood-ACK	0 s	50 s	35 s	2.5	N/A	N/A	N/A	2	50 s
GDflood-ACK	0 s	50 s	35 s	2.5	N/A	−1	1	2	80 s
NC-Dflood	0 s	70 s	30 s	$2.5{\cdot}g$	15	N/A	N/A	N/A	N/A
NC-GDflood	0 s	70 s	30 s	$2.5{\cdot}g$	15	0	1	N/A	N/A

or end-to-end ACKs. We have assumed this protocol due to its simplicity and suitability in today's underwater sensor networks, which are generally characterized by low traffic rates. Also a carrier sensing may be proposed for a short time before transmitting; however, it will not prevent all collisions. We leave this aspect for future work. Here, we have assumed a simple interference model. A total destructive interference is considered if two or more neighboring nodes are transmitting in the same time interval. This time interval consists of the time it takes for the packet to travel through the medium (a sound speed of 1500 m/s is assumed) plus the time required to receive the full packet. None of these nodes will corectly receive the packet intended for them. Also, the nodes that are neighbors, with the two overlapping transmitting nodes, will not receive any of these packets. To illustrate this, with reference to Fig. 3, let us consider that only the nodes 1 and 2 are transmitting in the same time interval. According to the considered interference model, the nodes 1, 2, and 7 will not receive correctly the packet, the transmission to the nodes 3 and 8 will not be affected by any interference. We have made these assumptions about the interference model in order to simulate the *worst case* scenario. In a real scenario, the performance will be better than the simulated one, because the interference is not always destructive. In some cases, the forward error correction bits can recover erroneous ones. In some other cases, it may happen that the channel buffers the packets from different nodes, and thus avoids the interference from different ongoing transmissions. With the latter assumption for the interference model, in the NC case, we have assumed that the network layer sends the packets down to the MAC layer with a backoff time drawn uniformly from [1, 10]. This is done with the goal to avoid interference that may affect all the packets of the same generation, if they are transmitted sequentially.

4.2 Simulation results

To evaluate the performance, we have considered three evaluation criteria: the PDR, the end-to-end delay, and the average number of packets forwarded by the network (Av. PKT), for each information packet produced by the source node. The protocol parameters are selected heuristically by a parameter scanning, as a trade-off between the three evaluation criteria, paying most attention to the PDR. The selected parameters are shown in Table 1. We first determined the Dflood parameters by simulations, considering $p = 0.1$ and $r = 1$ bit per second. The backoff time interval is the one that mostly influences the PDR and end-to-end delay. These parameters (i.e., $T_{\min}$ and $T_{\max}$) are selected in order to ensure a high value of the PDR (close to the saturation). The duplicate parameters, T_{dupl} and N_{dupl}, are selected in such a way to keep the PDR as high as possible and to reduce the average number of packets. These parameters have less influence on the PDR and end-to-end delay, compared to $T_{\min}$ and $T_{\max}$. For more details on the sensitivity to the three evaluation criteria, see [23]. Using the same approach, we find also the parameters for the other protocols.

For the GDflood protocol, we have used the same parameters as for Dflood[3], and an inaccuracy is considered when the distance D is measured. We have assumed

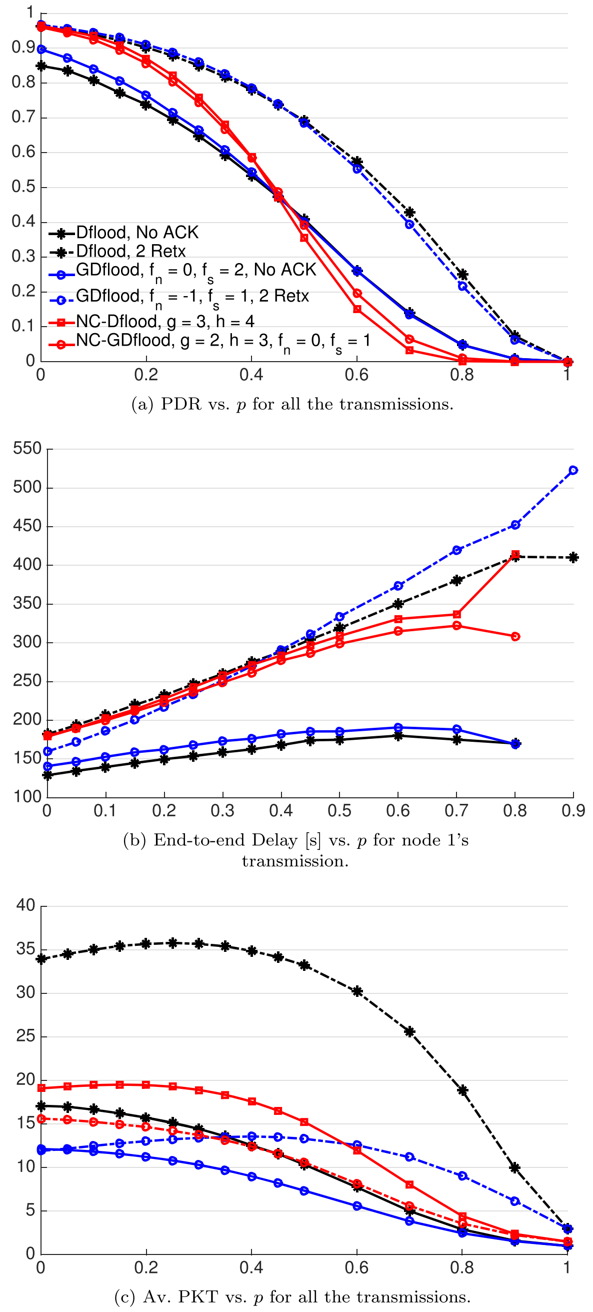
$$D = D^{\text{real}} + u, \tag{16}$$

where u is an error uniformly distributed in [−100, 100] m, and D^{real} is the real distance to the destination. When the implicit ACK is not used, the "best" parameters for GDflood are $f_{\text{n}} = 0$ and $f_{\text{s}} = 2$. From simulation results, we have found that, when we use the implicit ACK strategy, the "best" values are $f_{\text{n}} = -1$ and $f_{\text{s}} = 1$. In Dflood and GDflood, a good trade-off occurs when a maximum of two retransmissions is considered with t_{delay} equal to 50 and 80 s, respectively.

In the case of NC-Dflood, different values of the parameters with respect to Dflood give a better performance. More specifically, $T_{\min}$ and $T_{\max}$ are considered 0 s and 70 s respectively; a value of 30 s added only to the first scheduled packet is considered for T_{dupl}; for N_{dupl} the trade-off value is $2.5 \cdot g$ common to all the nodes in the network; and $H_{\max}$ is selected to be equal to 15. The value of $H_{\max}$ is found using the same approach as used for T_{dupl} and N_{dupl}. These values are selected for a generation size $g = 2$ and $h = 3$. It is obvious that for a larger g, the performance improves, up to a certain point, but the end-to-end delay and the Av. PKT increases as well. This is because, for larger generation sizes, the destination needs more packets to retrieve the information, which effectively increases the end-to-end delay. Also, larger generations cause higher energy consumption since less non-innovative packets are received at the relays, and if we try to reduce also the innovative information, the PDR will be affected. Meanwhile, an extra packet, during the encoding by the source nodes, is enough to improve the PDR without having big consequences for the Av. PKT. For higher values, the PDR starts saturating, and the energy consumption increases. In case the geographical position is used, we have found $f_\mathrm{n} = 0$ and $f_\mathrm{s} = 1$, respectively, as the best values keeping the other parameters same as in NC-Dflood.

Simulation results for different values of PER p and the traffic rate r are shown in Figs. 4 and 5. First, we have analysed the performance of our approaches and Dflood for different PERs, p. In this case, the traffic rate r of the application layer is considered 1 bit per second. In order to save space, we have shown the average PDR of all the transmissions, the end-to-end delay of the transmission of node 1 (considering that it is the farthest from the destination), and the mean of the Av. PKTs for the three singular transmissions in Fig 4a–c, respectively. From the first figure, we can see that Dflood without implicit ACK has the worst performance for a p smaller than 0.5. For reasonable small values of p, we can see that the retransmission strategy and NC offers values of the PDR that are close to the maximum. The GDflood protocol, even without implicit ACK, has a better PDR than Dflood since it reduces the number of collisions. The drawback is an increment of the end-to-end delay per packet, compared to the latter. As we can see from Fig. 4b, all the approaches result in a higher end-to-end delay compared to Dflood. In the case when the implicit ACK is used, this is obvious since the increment of the PDR is due to the retransmission strategy, which retransmits copies after a certain time. In the NC case, the selected parameters, which ensure the desired PDR value, also result in a higher end-to-end delay per packet. In Fig. 4c, we can see that the use of the implicit ACK in the Dflood protocol leads to a waste of energy, considering the high number of packets that are forwarded for each original one. Meanwhile, in the case of GDflood, the use of the retransmission strategy is more energy efficient. The use of the geographical position reduces the number of nodes that take part in the forwarding, and thus protocols that make use of it (including also NC-GDflood) flood fewer packets. For low values of p, we can reach values that are even lower than the pure Dflood protocol.

Without getting too much into detail, we can see that the same trend is kept also for different values of r, see Fig. 5a–c. In these simulations, we have assumed $p = 0.1$. We can see that the implicit ACK strategy improves the PDR, but at the same time a higher end-to-end delay is obtained. For the Dflood protocol, the implicit ACK is also worse in terms of energy, since the network floods too many packets. The use of the implicit ACK becomes more useful when the traffic increases, and thus the interference becomes more predominant. In those cases, by allowing a retransmission, we increase the possibility that a packet

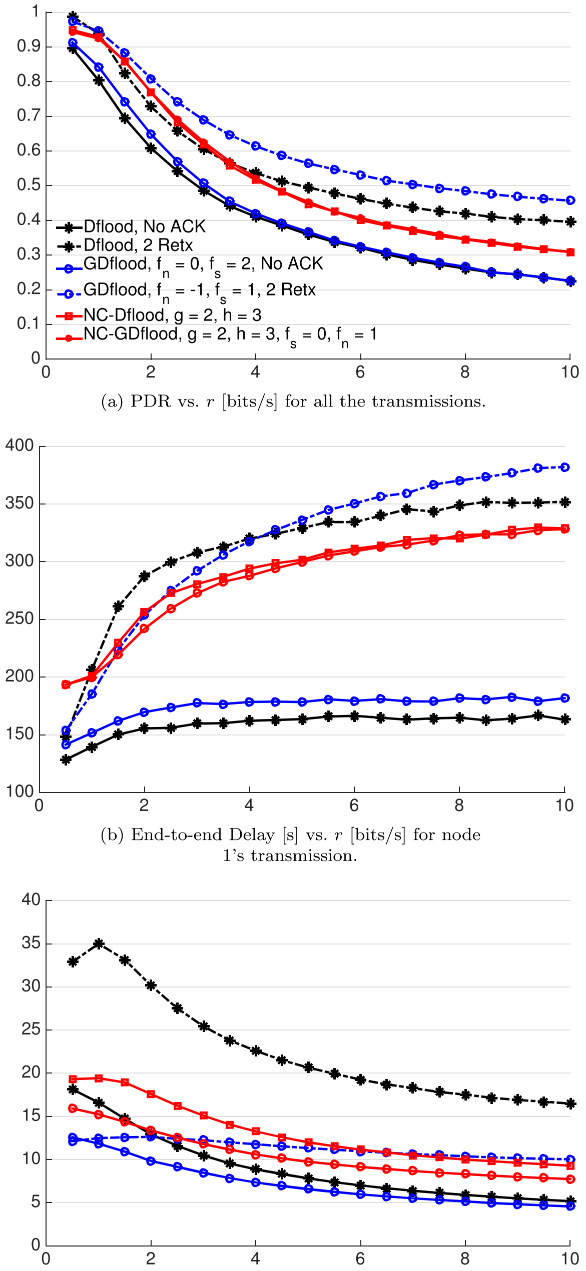


(a) PDR vs. p for all the transmissions.

(b) End-to-end Delay [s] vs. p for node 1's transmission.

(c) Av. PKT vs. p for all the transmissions.

Fig. 4 Performance comparison of proposed protocols with Dflood for different values of packet error rate, p. **a** PDR vs. p for all the transmissions. **b** End-to-end Delay [s] vs. p for node 1's transmission. **c** Av. PKT vs. p for all the transmissions



(a) PDR vs. r [bits/s] for all the transmissions.

(b) End-to-end Delay [s] vs. r [bits/s] for node 1's transmission.

(c) Av. PKT vs. r [bits/s] for all the transmissions.

Fig. 5 Performance comparison of proposed protocols with Dflood for different values of the traffic rate, r. **a** PDR vs. r [bits/s] for all the transmissions. **b** End-to-end Delay [s] vs. r [bits/s] for node 1's transmission. **c** Av. PKT vs. r [bits/s] for all the transmissions

reaches the destination. The NC approach still ensures high PDRs, especially for low traffic rates, matching those of Dflood and GDflood with the implicit ACK. It also ensures the same end-to-end delay, but more packets are transmitted compared to GDflood. It is worth to mention that the use of geographical information does not improve the PDR when the traffic increases. It seems that the limited number of relay nodes selected by GDflood is affected by the high interference present in the network. As we can see from Fig. 5a, when the traffic increases, and thus more collisions happen, all the protocols' performances degrade. This has direct implications on the end-to-end delay and average number of packets forwarded, as shown in Fig. 5b, c, respectively. However, most of today's underwater networks are characterized by sparse traffic, where the proposed protocols perform better.

4.3 Discussion

In evaluating the performance of the proposed protocols, we have been relying on simulation results. This is because the main goal of the paper is to propose to the reader an alternative way of using flooding in underwater sensor networks, and to illustrate its benefits in a basic scenario. However, in this subsection we discuss some challenges of the proposed approaches from a practical point of view.

For the GDflood and NC-GDflood, we consider that each node must know its own position and the final destination position. In underwater sensor networks, precise positioning is a challenge, especially for moving nodes [25]. However, our approach does not need precise information about the nodes' positions. Indeed, this information is only required to calculate the distance in hop counts between the transmitting node and the final destination. Considering that the position accuracy of such an estimate is about some tens of meters, or even hundreds of meters, and the transmitting distance is several kilometres, in the worst case, the D_{HC} will deviate by 1 from the true value. For this reason, in (16), we also consider an error on the true distance between the transmitting node and final destination.

Our next point considers the encoding coefficients required to perform the linear random NC, which must be transmitted alongside with the encoded packets. We want to point out that the benefits that NC brings with respect to Dflood (specially for low PER p and traffic rate r) come with a low extra information in the packet header. Indeed, the improved performance is achieved by only considering pairs of packets (i.e., $g = 2$) in the encoding process[4]. Thus, we may conclude that the use of NC is beneficial with a relative low price to pay in a flooding fashion.

From an implementation point of view, our protocols require a lot of data to be stored, e.g., the protocol's parameters, the already forwarded packets, the packets needed for applying NC and so on. In all these considerations, we assume that there are no memory constraints at the relay nodes. This is because most underwater sensors are bigger than terrestrial ones, and there is enough space to include a large memory.

Another point is the dependence of the protocols' parameters on the network topology. In this work, we consider a simple network topology, which satisfies the RACUN project requirements, mainly to illustrate that the proposed protocols might improve the performance of flooding-based protocols. However, we find it difficult to find a closed-form relationship between the protocols' parameters and the underlying topology, and thus we found their value with a scanning approach by simulations. It is obvious that for a different topology, their optimal value is different in order to ensure the highest performance. Indeed, they have to be tuned to the particular topology in order to achieve the desired performance. A challenging topology results when the AUV is the destination node. Geography-based protocols may suffer since the destination position changes continuously over time. In this case, the nodes may be informed about the operating sector of the AUV, and consider a reference point to calculate D_{HC}, e.g., this might be the center of the sector, or it can be improved if the AUV trajectory is know to the transmitting nodes. For the other protocols, this aspect is irrelevant as long as the AUV stays in the network coverage area.

5 Conclusions

In this work, we have proposed some advanced upgrades of a flooding-based protocol (Dflood), for underwater sensor networks. Our first idea was to incorporate the node position information in the relaying process. In this way, the participation of the nodes farther from the destination can be avoided. Simulation results show that a considerable amount of energy can be saved and an improvement of the PDR can be achieved as well. The price to pay is the end-to-end delay, which increases with respect to the original protocol. The use of an implicit ACK ensures that the PDR remains closer to the maximum for low values of the packet error rate, but our geographic approach outperforms the standard one in terms of energy consumption.

The second idea was to use network coding in order to flood more informative packets in the network than replicas. Considering the high amount of packets that are flooded in the network, the sharing of information between them will bring more information to the destination. Our proposed protocol, NC-Dflood, increases the PDR of the transmission with respect to Dflood (no ACK), with an increment in the end-to-end delay and energy consumption in a single transmission. An increment in the PDR, in this case, means that less original packets will be retransmitted in a second try in order to deliver all the information to the destination. Thus, the gap in the end-to-end delay and energy consumption is potentially reduced with respect to Dflood. Finally, we have used this approach also with geographical information and better results are obtained, even for a single transmission.

As future work, we consider the implementation of these ideas in real scenarios, with real communication channels, and different MAC protocols. We will also consider the implicit ACK strategy in the network coding case. Lastly, inspired by terrestrial wireless network coding schemes, it will be very interesting to use instantly-decodable random network coding [26, 27] in order to retrieve the original information packet before completing the full rank decodable problem.

Endnotes

[1] One more rule must be added to the rules in [9], which was inadvertently left out: "Forwarding is delayed by a time T_{dupl}, when a duplicate is received (with hop count greater than that of the original reception)" [10].

[2] Sometimes, the innovative packets are defined as those which increase the degrees of freedom, since they bring new information to that node.

[3] This is done in order to compare the GDflood performance with the "best" performance of Dflood. The protocol's parameters can be tuned to offer a better performance. Here, we have played only with f_{n} and f_{s} to achieve the desired performance of GDflood.

[4] Note that the protocol requires 1 extra byte for each encoded packet to be included in the packet header.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

The authors kindly acknowledge Paolo Casari (IMDEA Networks, Madrid, Spain) and Roald Otnes (FFI, Horten, Norway) for their support. This work has been supported by the European Defence Agency (EDA) project RACUN [28]. Part of this work has been presented in [29].

Author details

[1] Faculty of Electical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD, Delft, The Netherlands. [2] Acoustics and Sonar Dept., TNO, 2597 AK, The Hague, The Netherlands.

Received: 20 May 2015 Accepted: 6 April 2016
Published online: 03 May 2016

References

1. A Quasi, W Konrad, Underwater Acoustic Communications. IEEE Commun. Mag. **20**, 24–30 (1982)
2. IF Akyildiz, D Pompili, T Melodia, Underwater Acoustic Sensor Networks: Research Challenges. Ad Hoc Netw. Elsevier. **3**, 257–279 (2005)
3. R Otnes, et al., Underwater Acoustic Networking Techniques, Springer Briefs in Electrical and Computer Engineering (2012). doi:10.1007/978-3-642-25224-2-1
4. M Stojanovic, JC Preisig, Underwater Acoustic Communication Channels: Propagation Models and Statistical Characterization. IEEE Commun. Mag. **47**, 84–89 (2009)
5. R Urick, *Principles of Underwater Sound*. (McGraw-Hill, 1983)
6. EM Sozer, M Stojanovic, JG Proakis, Underwater Acoustic Networks. IEEE J. Oceanic Eng. **25**, 72–83 (2000)
7. G Han, J Jiang, N Bao, L Wan, M Guizani, Routing Protocols for Underwater Wireless Sensor Networks. IEEE Commun. Mag. **53**, 72–78 (2015)
8. RWL Coutinho, A Boukerche, LFM Vieira, AAF Loureiro, Design Guidelines for Opportunistic Routing in Underwater Networks. IEEE Commun. Mag. **54**, 40–48 (2016)
9. R Otnes, S Haavik, *Duplicate Reduction with Adaptive Backoff for a Flooding-Based Underwater Network Protocol*. (IEEE Oceans, Norway, 2013)

10. R Otnes, PA van Walree, H Buen, H Song, Underwater Acoustic Network Simulation with Lookup Tables from Physical-Layer Replay. IEEE J. Oceanic Eng. **40**, 822–840 (2015)
11. P Xie, JH Cui, L Lao, VBF, in *Proc. of IFPF Networking*. Vector-Based Forwarding Protocol for Underwater Sensor Networks, (Canada, 2005)
12. JM Jornet, M Stojanovic, M Zorzi, On Joint Frequency and Power Allocation in a Cross-Layer Protocol for Underwater Acoustic Networks. IEEE J. Oceanic Eng. **35**, 936–947 (2010)
13. R Ahlswede, N Cai, SR Li, RW Yeng, Network Information Flow. IEEE Trans. Information Theory. **46**, 1204–1216 (2000)
14. Z Guo, P Xie, JH Cui, B Wang, *On applying Network Coding to Underwater Sensor Networks*. (WUWNET'06, LA California USA, 2006)
15. DE Lucani, M Medrad, M Stojanovic, *Network Coding Schemes for Underwater Networks*. (WUWNET'07, Canada, 2007)
16. N Chirdchoo, M Chitre, WS Soh, *A Study on Network Coding in Underwater Networks*. (IEEE Oceans, Washington USA, 2010)
17. Z Guo, B Wang, P Xie, W Zeng, JH Cui, Efficient Error Recovery with Network Coding in Underwater Sensor Networks, Ad Hoc Networks, Elsevier. **7**, 791–802 (2009)
18. V Kebkal, K Kebkal, O Kebkal, *Experiments with Network Coding in Dynamic Underwater Acoustic Channel*. (IEEE UComms, Italy, 2014)
19. C Meanville, A Miyajan, A Alharbi, M Haining, *Network Coding in Underwater Sensor Networks*. (IEEE Oceans, Norway, 2013)
20. C Fragouli, JY Le Boudec, J Widmer, Network Coding: An Instant Primer, LCA-Report-2005-010 (2005). Available online from, www.infospace.epfl.ch
21. PA Chou, Y Wu, K Jain, in *Proc. of Allerton Conference on Communication*. Practical Network Coding (Control and Computing, Illinois USA, p. 2003
22. T Ho, R Koetter, M Medrad, DR Krager, M Effors, in *International Symposium on Information Theory (ISIT)*. The Benefits of Coding Over Routing in a Randomized Setting, Japan, 2003)
23. E Isufi, *Network Coding for Flooding-Based Routing in Underwater Sensor Networks, Master Thesis*. (University of Perugia, Italy, 2014)
24. S Basagni, C Petrioli, R Petroccia, M Stojanovic, Optimized Packet Size Selection in Underwater Wireless Sensor Network Communications. IEEE J. Oceanic Eng. **37**, 321–337 (2012)
25. H Yan, Z Shi, JH Cui, *DBR: Depth-Based Routing for Underwater Sensor Networks*. (IFIP Networking'08, Singapore, 2008)
26. Y Mingchao, P Sadeghi, N Aboutorab, Performance characterization and transmission schemes for instantly decodable network coding in wireless broadcast. EURASIP J. Adv. Signal Process. **2015**, 1–17 (2015)
27. A Douik, S Sorour, TY Al-Naffouri, MS Alouini, Instantly Decodable Network Coding for Real-Time Device-to-Device Communications. EURASIP J. Adv. Signal Process. **2016**, 1–14 (2016)
28. J Kalwa, *The RACUN-Project: Robust Acoustic Communications in Underwater Networks - an Overview*. (IEEE Oceans, Spain, 2011)
29. E Isufi, G Leus, H Dol, in *Proceedings of the International Conference on Underwater Networks and Systems (WUWNET'14)*. Network Coding for Flooding-Based Routing in Underwater Sensor Networks (ACM, Italy, 2014). http://dl.acm.org/citation.cfm?id=2674570&CFID=600486126&CFTOKEN=90052992. doi:10.1145/2671490.2674570