

An FEC-based Reliable Data Transport Protocol for Underwater Sensor Networks

Peng Xie and Jun-Hong Cui
 xp@engr.uconn.edu, jcui@cse.uconn.edu
 Computer Science and Engineering Department
 University of Connecticut, Storrs, CT 06269-2155

Abstract—In this paper, we investigate the reliable data transport problem in underwater sensor networks. Underwater sensor networks are significantly different from terrestrial sensor networks in two aspects: acoustic channels are used for communication and most sensor nodes are mobile due to water current. These distinctions feature underwater sensor networks with low bandwidth capacity, large propagation delay, high error probability, half-duplex channels, and highly dynamic topology, which pose many new challenges for reliable data transport in underwater sensor networks. In this paper, we propose a protocol, called segmented data reliable transport (SDRT), to achieve reliable data transfer in underwater sensor networks. SDRT is essentially a hybrid approach of ARQ and FEC. It adopts efficient erasure codes (so-called SVT codes in this paper), transferring encoded packets block by block and hop by hop. Compared with other existing reliable data transport approaches for underwater networks, SDRT can reduce the total number of transmitted packets, improve channel utilization, and simplify protocol management. In addition, we develop a mathematic model to estimate the expected number of packets actually needed. Based on this model, we can set the block size appropriately for SDRT, as helps to address the node mobility issue. We conduct simulations to evaluate our model and SDRT. The results show that our model can closely predict the number of packets actually needed, and SDRT is energy efficient and can achieve high channel utilization.

I. INTRODUCTION

Underwater sensor networks have received growing interests [19], [13], [1], [8], [6] recently. Compared with traditional techniques (such as remote sensing) used in aquatic applications, underwater sensor networks empower us to monitor/detect phenomena more accurately and timely in extended areas.

As in terrestrial sensor networks, for mission critical applications, reliable data transport is demanded in underwater sensor networks. To design a good reliable data transport protocol, a natural concern is the energy efficiency issue since, similar to terrestrial sensor nodes, underwater sensor nodes are also usually powered by batteries. In harsh aquatic environments, it is even more difficult to recharge or replace batteries. Besides energy consumption, there are several other important issues to consider due to the significant distinctions between underwater sensor networks and terrestrial sensor networks, which we describe as follows.

First, **acoustic channels are used for underwater communication**. In underwater environments, radio does not

work well due to extremely limited propagation distance, thus acoustic channels are employed. The propagation speed of acoustic signals in water is about 1.5×10^3 m/s, five orders of magnitude lower than the radio propagation speed (3×10^8 m/s). Moreover, underwater acoustic channels are affected by many factors such as path loss, noise, multi-path, and Doppler spread. All these cause high error probability in acoustic channels. Further, due to mechanical limitations (sound is a mechanical wave), acoustic modems typically operate in half-duplex, and the transition between sending and receiving modes is very slow, usually taking tens of milliseconds [7].

Second, **underwater sensor nodes are passively mobile**. Empirical observation suggests that water current move at the speed of 3-6 kilometers per hour in a typical underwater condition [2]. In an underwater sensor network, majority of the sensor nodes (except some fixed nodes equipped on surface-level buoys) are mobile due to water currents¹. This kind of node mobility results in highly network topology dynamics.

In summary, underwater sensor networks have the following unique characteristics: large propagation delay, low bandwidth capacity, half-duplex channels, high error probability, and high topology dynamics. All these features pose many challenges for reliable data transport in underwater sensor networks.

A. Design Challenges

The new characteristics discussed above make many common wisdoms in reliable data transport undesirable for underwater sensor networks.

1) *End-to-End approach does not work well for underwater sensor networks*: Broadly speaking, there are two types of approaches for reliable data transport, namely, end-to-end and hop-by-hop. Some recent studies [18], [9], [15] show that the end-to-end approach is infeasible for sensor networks. This conclusion still holds in underwater sensor networks, and is additionally justified by the high channel error probability and the low propagation speed of acoustic signals: (1) The high channel error probability makes the probability of successfully transferring data from end to end almost approaching to

¹In this paper, we mainly discuss “mobile” underwater sensor networks, where sensors are not fixed and can passively float with water currents. This feature is common in a wide range of aquatic applications, such as estuary monitoring and submarine detection [6]. However, our proposed approach can be easily adapted to stationary underwater sensor networks.

*This work is supported in part by the NSF CAREER Grant No. 0644190.

0, as results in too many retransmissions for a successful packet delivery; (2) The low propagation speed of acoustic signals (1500 m/s) will cause very large end-to-end delay, which introduces difficulty for the two ends to manage data transmission timely.

2) *Half-Duplex acoustic channels limit the choice of complex ARQ protocols:* Many reliable data transfer protocols use ARQ (Automatic Repeat Request), in which the receiver needs to send acknowledgement (ACK) to the sender, requesting for retransmission if packets are lost. The simplest ARQ protocol is the Stop-and-Wait protocol. In this protocol, after the sender sends one packet, it waits for ACK from the receiver. Only when a positive ACK is received, it starts to send a new packet. It is well-known that the channel utilization of this simple protocol is very low, especially when the signal propagation delay is large. Pipelined ARQ protocols, such as Go-Back-N and Selective Repeat, can significantly improve channel utilization. However, half-duplex acoustic channels limit the choice of these complex ARQ protocols which require full-duplex operation.

3) *Too many feedback from receivers are not desirable:* In the literature, there are several improved versions of Stop-and-Wait protocols, which are designed to increase channel utilization [11], [17]. In such protocols with many ACKs, if ACKs are lost, not only bandwidth resource is wasted for ACK transmission, but also some successfully received packets will be retransmitted by the sender, causing more energy consumption. Additionally, in a densely deployed sensor network, multiple sensor nodes in the transmission range of the sender can overhear the transmission of the same packets. In this scenario, if one receiver requests for retransmission of lost packets, other receivers have to consume energy in overhearing the duplicate packets.

4) *Pure Feedback-Free protocols are not energy efficient:* Some reliable data transport protocols resort to Forward-Error-Correcting (FEC) to overcome the problems caused by many ACKs. In an FEC approach, packets are encoded at the sender, and then transmitted continuously to the receiver. At the receiver side, lost packets are ignored, and original data packets can be reconstructed after enough number of encoded packets are successfully received. The management of feedback-free protocols using FEC is relatively simple (compared with ARQ protocols, which usually involve complex timeout and synchronization management) in that encoding and decoding are the only additional overhead introduced to the sender and receiver respectively. However, FEC essentially exploits redundancy for reliability. In other words, additional energy is wasted to achieve reliable data transport. In sensor networks, due to energy constraints, pure FEC protocols may not be a good choice.

5) *Very large bulk data transmission is not suitable in mobile underwater sensor networks:* In our target underwater sensor network scenario, most nodes are mobile. Therefore, there is no long-lasting fixed neighborhood, i.e., the communication time between any pair of sender and receiver is limited. Moreover, the bandwidth of communication channels is relatively low, and the propagation delay is extremely high. All these factors restrict the sender from sending very large

bulk data at one time.

B. SDRT Overview and Contributions

Summarizing the above discussions, we aim to design a reliable data transport protocol for underwater sensor networks, with the goals of *high energy efficiency, high channel utilization and simple protocol management*. In this paper, we propose a protocol, called **Segmented Data Reliable Transport (SDRT)**, which is essentially a hybrid approach exploring both FEC and ARQ. SDRT assumes that receivers can detect corrupted packets. When a packet is completely lost or unrecoverable from corruption, this packet is treated as “lost”. In this paper, we are interested in reconstructing lost packets, not error-correction within packets.

In SDRT, a data source node first groups data packets into blocks. Then the data packets are delivered from the source to the destination, block by block and hop-by-hop². An intermediate node encodes each data block using an efficient erasure coding scheme (SDRT adopts a simple variant of Tornado codes [10], referred to as **SVT** codes, which will be discussed in Section II) and pumps encoded packets into the channel. After a receiver receives the encoded packets, it decodes and reconstructs the original block. After the reconstruction is done, the receiver encodes the block again and relays the block to the nodes in next hop. For each relay of a data block, the sender keeps pumping encoded packets until receiving a positive ACK from its next hop.

Our contributions can be summarized are follows:

- SDRT can well satisfy the harsh requirements in underwater sensor networks: efficient SVT codes enable SDRT to significantly improve channel utilization and relieve both senders and receivers of handling many ACKs; more important, SDRT reduces the number of packets to send, thus saving energy.
- SDRT also helps to address the dynamic network topology problem caused by sensor node passive mobility. By setting an appropriate block size, SDRT controls the transmission time of each block, thus, loosening the requirements for the routing protocol to select next hop. The central issue for setting block size is to estimate the number of packets actually needed to recover the original data packets. We develop a mathematical model, which can help to set the block size appropriately.
- We conduct simulations to evaluate our model and SDRT. The results show that our model can closely estimate the expected number of needed packets, and SDRT is energy efficient, and can achieve high channel utilization.

The rest of this paper is organized as follows. We first give a brief review of SVT codes in Section II. Then we present the SDRT protocol in Section III. After that, we describe the mathematical model of estimating the expected number of needed packets and the block size in Section IV-A, followed by the performance evaluation in Section V. Finally,

²In this paper, we design SDRT protocol using the hop-by-hop approach. However, it does not mean that there is no need to achieve end-to-end reliability. In fact, in the case of network disconnection or congestion, some mechanism should be incorporated to provide end-to-end feedback, achieving the ultimate end-to-end reliability.

we discuss several seminal proposals for reliable data transport in sensor networks in Section VI, and conclude our paper in Section VII.

II. SIMPLE VARIANT OF TORNADO (SVT) CODES

Since SVT codes are simplified Tornado codes, we will first introduce Tornado codes in the following.

Tornado codes [10] are one type of erasure codes, which typically encode a set of original n messages into a set of N encoded messages, where $N \geq n$. In order to reconstruct the n original messages, the receiver has to receive a certain number (larger than n) of encoded messages. The stretch factor, defined as $\frac{N}{n}$, is used to measure the redundancy of erasure codes. In Tornado codes, encoding and decoding only involve XOR operations, and thus are usually faster than other coding methods, such as Reed-Solomon codes, which require computation-intensive field operations [14]. Before describing the encoding and decoding algorithms of Tornado codes, we give some definitions. We call the original n messages as data packets, all the other messages are called check packets. In this paper, we use “packet” to refer to a data packet or a check packet. XOR operation is denoted as \oplus . If we use P_1 and P_2 to represent two packets of the same size, then $P_1 \oplus P_2$ is the result of bitwise-XOR’ing packets P_1 and P_2 .

Encoding: The encoding algorithm in Tornado codes is equivalent to constructing a multi-level bipartite graph where the nodes in each level are only randomly connected to the nodes in adjacent levels. In the graph, the nodes in the leftmost level denote original data packets, and the nodes in each other level are the XORs of all their neighbors (nodes) in the left adjacent level. The randomly ordered packets including both data packets and check packets constitute the final encoded packets. An example of three-level Tornado codes is shown in Fig. 1. In this figure, d_1, d_2, \dots, d_5 denote the original data packets. $c_1 = d_1 \oplus d_2 \oplus d_3$, $c_2 = d_1 \oplus d_3 \oplus d_5$, $c_3 = d_2 \oplus d_4$ are the nodes in the second level, and $b_1 = c_1 \oplus c_2$ and $b_2 = c_1 \oplus c_3$ constitute the third level.

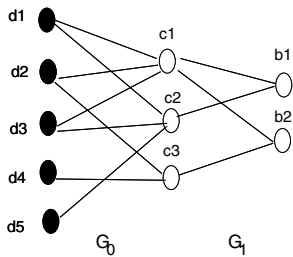


Fig. 1. An example of Tornado codes.

For a multiple-level bipartite graph, any two adjacent levels are connected by edges and constitute a *subgraph*, denoted as G_k , which specifies the way to generate the check packets. As shown in Fig. 1, G_0 and G_1 are two subgraphs. In a *subgraph*, an edge is an *edge of degree i* on the left (right) if it is adjacent to a node of degree i on the right (left). For example, in subgraph G_0 , edges adjacent to node d_1 are edge of degree 2 on the left since node d_1 is a node with degree 2 on the right. A subgraph is determined by two edge-degree vectors,

namely, left degree vector $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_M)$ and right degree vector $\rho = (\rho_1, \rho_2, \dots, \rho_M)$, where λ_i is the fraction of edges with the degree i on the left and ρ_i the fraction of edges with the degree of i on the right. Thus for subgraph G_0 , $\lambda = (\frac{2}{8}, \frac{6}{8})$ and $\rho = (0, \frac{2}{8}, \frac{6}{8})$. Given these two vectors, we can generate the subgraph by a method suggested in [10].

Decoding: The encoding process of Tornado codes is to construct a graph from left to right, while the decoding process of Tornado codes is to reconstruct the graph from the right level to the left level. In a subgraph G_k , if most of the nodes are known, then we have a higher chance to recursively reconstruct the unknown nodes until all the nodes in the left level (which is the right level for the subgraph G_{k-1}) are recovered. Following this way, we can recover all the nodes till the leftmost level are reconstructed. The reconstruction processes within each subgraph is one step of the decoding process. In each decoding step, for a node on the right side, if all but one of its neighbors are known, then this lost packet can be recovered by XORing all the related packets. For example, in Fig. 1, suppose packet d_1 is lost, we can reconstruct d_1 by $d_2 \oplus d_3 \oplus c_1$. For a subgraph G_k , its *decoding subgraph* consists of the nodes on the left side that are lost and not decoded yet, all nodes on the right side (already recovered in subgraph G_{k+1}) and all the edges between them. One decoding step is equivalent to finding a node of degree 1 on the right side in the *decoding subgraph*, and removing this node, its left neighbor nodes and all edges adjacent to its neighbors from this *decoding subgraph*. This procedure is repeated until all the lost packets are reconstructed or no more node of degree 1 on the right side.

SVT Codes: Tornado codes are preferred in many network applications in that they can be easily and fast encoded and decoded [5], [4], [3]. However, the degree used in the original design of Tornado codes, which is derived from differential equations and relatively large, is not suitable for our scenarios. Since the number of packets in each block is very small, it is hard to find the optimal degree vectors required by the original Tornado codes. Moreover, a large degree results in more packet header overhead. Thus, we adopt a simple variant of Tornado codes (referred to as **SVT** codes in this paper) for SDRT. SVT codes are essentially two-layer Tornado codes, with the first two elements of the left degree vector λ fixed at 0 (i.e., $\lambda_1 = \lambda_2 = 0$). The rationale behind this choice of degree distribution is following: in [10], it is proved that, to make the decoding of Tornado codes efficient, the left degrees of nodes are at least 3, that is, $\lambda_1 = \lambda_2 = 0$.

III. SDRT PROTOCOL DESIGN

A. Network Setting

Our target sensor network is relatively dense, with nodes working in low-power transmission range (less than $100m$). The available data rate in such a network is expected to be around 10kbps. In the network, majority of the nodes are mobile, except some surface gateway nodes which are attached to buoys. A data source node generates data packets and relays to its neighbor sensor nodes. Through multiple routing hops, the data packets finally reach surface gateways, which then transfer data to the onshore command center via radio.

B. Protocol Description

The key idea of the segmented data reliable transport (SDRT) protocol is to transfer encoded packets (using SVT codes), block by block and hop-by-hop. In order to reconstruct the original data packets, the receiver has to receive sufficient encoded packets. Because the node mobility in underwater environment results in short communication time between any pair of sender and receiver, the transmission time for the encoded packets is limited. Thus SDRT has to guarantee that the receiver can receive enough encoded packets in such a limited time interval. By setting the block size n (i.e., the number of original data packets in each block) appropriately, SDRT can control the transmission time and allow the receiver to be able to receive enough packets in order to reconstruct original block even in node motion.

In SDRT, a data source first groups data packets into blocks of size n . Then the source encodes these blocks of packets, and sends the encoded blocks into the network. The data packets are forwarded from the source to the destination block by block, and each block is forwarded hop-by-hop. In each hop-by-hop relay, the sender first estimates the number of packets it needs to send for the receiver to reconstruct the original packets (using the model presented in Section IV). We call this number as “sending window”. Within the sending window, the sender pumps the encoded packets to the network fast. When the window is reached, the sender slows down pack transmission, waiting for a positive feedback from the receiver³. After receiving encoded packets, the receiver tries to reconstruct the original data packets. If the reconstruction is successful, it sends back a positive feedback. Upon the reception of a feedback, the sender stops sending packets, while the receiver encodes the original data packets again and relays them to the next hop.

The operations performed on the sender and the receiver are described in the following.

SDRT Sender:

- 1) *Estimates the sending window.*
- 2) *Encodes a block using SVT codes.*
- 3) *Pumps encoded packets fast (in a random order) within the sending window.*
- 4) *Sends encodes packets slowly outside the sending window until receiving a positive feedback from the receiver.*

SDRT Receiver:

- 1) *Keeps receiving packets until it can reconstruct the original data packets, and sends a positive feedback to the sender⁴.*
- 2) *Encodes the reconstructed packets again and relay them to the next hop.*

From the above description, we can see that SDRT reduces the burden of the sender and the receiver by requiring only one feedback per block. The sender has no additional responsibilities except encoding and injecting packets, and the receiver only needs to send one feedback after reconstructing the original packets.

³The slow sending rate outside the sending window allows the sender to switch to receiving to listen for the feedback from the receiver.

⁴In real implementation, the feedback can be sent several times to make sure that the sender can successfully receive a feedback.

C. Discussions

1) *Performance Improvement:* In SDRT, a sender keeps sending randomly ordered encoded packets into the network until receiving a positive feedback for each block. On the one hand, this improves the channel utilization since the sender does not need to wait for the feedback from the receiver before transmitting subsequent packets. On the other hand, as will be shown in Section V, SDRT actually reduces the number of packets transmitted, therefore, saves more energy.

2) *Multiple Receivers:* In underwater sensor networks, it is common that there are multiple nodes in the transmission range of a sender. All these nodes can overhear the transmission of the same packets and they are potentially capable of relaying the block to the next hop. As pointed in [9], sometimes it is necessary for the routing protocol to provide alternative paths to overcome link failures. Our SDRT protocol can be smoothly integrated with the routing protocols that support alternative paths to improve the network robustness. Our theoretical analysis and simulation results will show that SDRT performs even better in the multiple-receiver case.

3) *Sensor Node Mobility:* In underwater sensor networks, the dynamic network topology issue is a major concern of routing protocols. SDRT contributes to address this problem by setting an appropriate block size such that the next hop has sufficient time to reconstruct the whole block even in motion. The block size n is determined by many factors such as the implementation of SVT codes, the mobile speed of nodes, the available bandwidth, the sound propagation speed and the distance between the sender and the receiver. We will present an approach to estimate the block size, n , in section IV-C.

IV. ANALYSIS

In this section, we present a model to estimate the expected number of needed packets, and discuss how to calculate the block size for SDRT.

A. Modelling SVT Codes

We have the following assumptions for the SVT codes we use. First, we assume packet losses are independent. Second, we assume that the sender only needs to send at most three rounds of encoded packets to guarantee that the receiver successfully recovers the original data packets. This can be achieved by selecting appropriate left degree vector and right degree vector. Our model can be easily extended to SVT codes with more than three rounds.

We use a bipartite graph with two levels to represent SVT codes. Let $G = \{V, E\}$ be the bipartite graph, where E is the set of edges and V is the set of nodes in the graph. $V = D \cup C$ and $D \cap C = \phi$, where D is the set of data packets and C is the set of check packets. The edges randomly connect the nodes in D and the nodes in C . For each node $v \in C$, we define a *box*, $b_v = \{u | u = v, \text{ or } uv \in E\}$. Thus, for each node $v \in C$, there is a corresponding box, which is the set of this node and its neighbors. There are totally $|C|$ boxes in graph G . If the left degree of node $v \in C$ is i , we say the degree of b_v is i and the capacity of b_v (i.e., the number of nodes in this box) is $i + 1$. We call the set of all the boxes with the same degree a *cluster*. The cluster with degree i is denoted as B_i .

For a sender-receiver pair, the receiving process at the receiver side is equivalent to filling packets into $|C|$ boxes. When a check packet is delivered successfully, it is equivalent to putting one packet into one box. If a data packet of degree j is received, it is equivalent to put j packets into j different boxes. For clarity, we refer the packets generated by a data packet or a check packet to as *replicas*. The replicas generated by the same data packet are called *sibling replicas*. For simplicity, we assume that sibling replicas are independent of each other. This assumption is reasonable to some extent since sibling replicas are independently and randomly put into different boxes. When a box of capacity k is filled with $k-1$ or k packets, we consider this box full, i.e., all the packets in this box can be recovered. Using SVT codes, one full box can potentially cause other boxes full.

We still use λ and ρ to specify the edge-degree vectors, i.e., $\lambda = \{\lambda_1, \lambda_2, \dots, \lambda_M\}$ is the left degree vector and $\rho = \{\rho_1, \rho_2, \dots, \rho_M\}$ is the right degree vector. We use p to denote the (packet) erasure probability, n to denote the number of original data packets or block size, and $1 + \beta$ to denote the stretch factor. It is easy to get $|D| = n$, $|C| = n\beta$, and $N = |C| + |D| = (1 + \beta)n$.

B. Estimating Expected Number of Needed Packets

In this subsection, we first examine the single receiver case and present a model to estimate the expected number of packets actually needed by the receiver to reconstruct the original data packets, then we extend the model for the case of multiple receivers.

1) *Single Receiver*: Let $f(x)$ be the probability that the receiver reconstructs the original data packets given that x packets has been sent out before. Let $d(1|x)$ be the probability that a successfully delivered packet is a duplicate one given that x packets has been sent out (by the sender) before, and $r(1|x)$ be the probability that the receiver can reconstruct the original data packets when a non-duplicate packet is successfully delivered given that x packets has been sent out before. Then, we have

$$\begin{aligned} f(x+1) &= pf(x) + (1-p)(d(1|x)f(x) \\ &\quad + (1-d(1|x))r(1|x)) \\ f(0) &= 0 \end{aligned} \quad (1)$$

Due to space limit, we will not include the procedures of evaluating $r(1|x)$ and $d(1|x)$ in this paper. Interested readers can refer to our technical report [20].

Now we can evaluate the probability that a receiver recovers the original data packets when the sender sends out exactly x encoded packets. Let $P(x)$ denote this probability, and it is computed as

$$P(x) = f(x) - f(x-1) \quad (2)$$

Then, the average number of packets a sender needs to send can be evaluated as

$$E_1 = \sum_{i=0}^{\infty} i \times P(i) \quad (3)$$

2) *Multiple Receivers*: Now we consider the case with R receivers, where $R \geq 2$. When the sender sends one packet, we assume that this packet is transmitted to R receivers independently.

For one receiver, we can use random variable X to denote the number of successfully received packets given that k packets have been sent out by the sender. We can represent X as $X = \sum_j X_j$, where random variable X_j has value 1 if a packet is successfully delivered, and 0 otherwise. Further, we have notations $P_r(X_j = 1) = 1 - p$ and $P_r(X_j = 0) = p$. Then, X has a binomial distribution, with expected value as $\mu = k(1-p)$ and standard deviation as $\delta = \sqrt{k \times (1-p) \times p}$.

For the multiple-receiver case, R receivers are equivalent to R trials of X with the same binomial distribution. If any one receiver can reconstruct the original data packets and thus send a positive feedback, the sender then stops sending packets. In other words, we need to find the expected maximum value among these R trials. We evaluate the expected maximum value of these R trials by the Chebyshev's Inequality [12]. The expected number of packets needed in the case of R receivers, E_R , can be evaluated as follows:

$$E_R = \frac{(\sqrt{Rp + 4pE_1} - \sqrt{Rp})^2}{4p}. \quad (4)$$

From Equation 4, we can easily prove that $E_R \leq E_1$. This means, in SDRT, the sender actually sends fewer packets when there are more than one receivers.

C. Calculating Block Size

As stated earlier, in order to address the node mobility issue in underwater sensor networks, SDRT has to properly set the block size, n . Block size is determined by many factors, such as the stretch factor, the mobile speed of the nodes, the bandwidth of communication channels and the propagation speed of sound. We use $1 + \beta$ to denote the stretch factor (same as before), bw to denote the bandwidth of communication channels, L to denote the packet size, V to be the propagation speed of acoustic signals and i be the number of receivers ($i=1$ or R). Further, we denote the possible minimum time interval for the communication between a sender and a receiver (supported by routing protocols) as T_r . Then the block size, n , must satisfy the following inequity:

$$\frac{E_i \times L}{bw} < T_r \quad (5)$$

Note that n is one parameter in computing E_i . Due to the complicated procedure of estimating E_i , we can only resort to numerical methods to compute n . It should be further noted that the block size estimation is conducted off-line, before the underwater sensor network deployment. Thus, the complexity of block size estimation will not burden sensor nodes.

An Example: If we have $n = 1000$, $\beta = 0.6$, $bw = 50$ kbps, $V = 1500$ m/s, the erasure probability $p = 0.5$, the packet size $L = 40$ bytes, the left degree vector $\lambda = \{0, 0, 0, 1\}$ and the right degree vector $\rho = \{0, 0, 0, 0, \frac{1}{8}, 0, \frac{7}{8}\}$, we can estimate the average number of needed packets, which is 4185 based on Equation 3, for the case of single receiver. Then according to Inequity 5, the transmission time is less than 26.2

seconds. In other words, a block size of 1000 requires that the minimum time interval for the communication between a sender and a receiver is at least 26.2 seconds. This means that the receiver should stay in the transmission range of the sender for at least 26.2 seconds. If the node transmission range is 100 m, then roughly speaking, the whole block of data can be successfully transmitted from the sender to the receiver, considering the node moving speed is less than 3 m/s.

V. PERFORMANCE EVALUATION

In this section, we evaluate the performance of SDRT by simulations. We have also conducted experiments to evaluate the accuracy of the packet estimation model proposed in Section IV. Due to space limit, we will not show the results for model evaluation in this paper. Interested readers can refer to our technical report [20].

A. Performance Metrics

We define the following metrics for SDRT.

- 1) Goodput: This metric is define as

$$\theta = \frac{\# \text{ of original data packets}}{\text{Time to successfully send all packets}}. \quad (6)$$

Goodput θ is a measure of channel utilization: the higher the goodput, the better the utilization, and vice versa.

- 2) Sending Inefficiency: This metric is defined as

$$\alpha = \frac{\# \text{ of packets sent}}{\# \text{ of original data packets}}. \quad (7)$$

We use sending inefficiency α to measure energy efficiency. The smaller the inefficiency, the fewer packets are sent; thus, less energy is consumed.

B. Results and Analysis

- 1) *Channel Utilization*:: First, we compare the goodput of SDRT with a naive ARQ approach and an accumulative ARQ approach, in the case of single receiver. The naive ARQ approach is similar to the basic Stop-and-Wait protocol, and the accumulative ARQ approach is essentially the same as the S&W-2 approach [11].

In this set of simulations, we fix the packet size as 40 bytes and the bandwidth as 10 kbps. The distance between the sender and the receiver is set to 60m and 90m. The SVT codes have non-uniform degree distribution: $\lambda = (0, 0, 0, 1)$ and $\rho = (0, 0, 0, 0, \frac{1}{8}, 0, \frac{7}{8})$. The simulations are run for 100 trials, and the final results are plotted in Fig. 2.

From Fig. 2, we can observe that the goodput of SDRT is two orders of magnitude higher than that of the naive ARQ and the accumulative ARQ, as indicates that the channel utilization is significantly improved in SDRT. If we compare the naive ARQ and the accumulative ARQ, the latter slightly improves the goodput. This can be explained as follows: (1) In the accumulative ARQ, the sender reduces time on waiting ACKs by sending packets and ACKs in burst; (2) The accumulative ARQ reduces redundant packet retransmissions (caused by lost ACKs in the naive ARQ) by using accumulative ACKs. This is consistent with the conclusion drawn in [16].

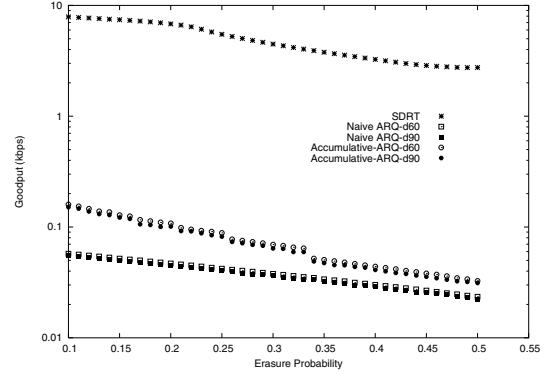


Fig. 2. Goodput vs. the erasure probability for single receiver, where -d60 means the case of 60m, and -d90 means the case of 90m.

Another observation from Fig. 2 is that the distance between the sender and the receiver has almost zero effect on the goodput of SDRT (thus in the figure, only one curve is shown). This is due to the fact that SDRT only needs a very limited number of ACKs. However, distance plays a non-negligible role in both the naive ARQ and the accumulative ARQ. This tells us that SDRT is more robust to network density.

- 2) *Energy Efficiency*: In this subsection, we compare the sending inefficiency of four protocols: data carousel, SDRT, naive ARQ, and accumulative ARQ. The naive ARQ and accumulative ARQ protocols are the same as discussed in the last subsection. The data carousel protocol is essentially a brute-force data transmission approach: no packet encoding/decoding, and no ACKs either. The sender simply keeps sending data packets in a random order until the receiver receives these data packets successfully.

In this set of simulations, the number of data packets is set to 1000. For SDRT, the block size is 1000, and the stretch factor is 1.6, i.e., there are 1000 data packets and 600 check packets in each block. We use the SVT codes with the left degree vector $(0, 0, 0, 1)$ and the right degree vector $(0, 0, 0, 0, \frac{1}{8}, 0, \frac{7}{8})$. Further, we fix the network size (with 20 hops), and vary the erasure probabilities from 0.1 to 0.5. We conduct simulations in both single-receiver and three-receiver scenarios. The results are shown in Fig. 3.

From Fig. 3, a general trend we can get is that when the erasure probability increases, the sending inefficiency is lifted, i.e., the energy efficiency is reduced. This is pretty intuitive: the poorer the channels, the more energy is needed to achieve reliability. Comparing the four protocols, SDRT has the least sending inefficiency, therefore, fewest packets are sent; thus the energy efficiency is the best, in all cases. In addition, SDRT and the data carousel protocol perform better in the case of three-receiver than in the case of single-receiver, as is consistent with the analysis in Section IV. However, this conclusion does not hold for the naive ARQ and accumulative ARQ, where the sender needs to send more packets when there are three receivers. This is mainly due to the following two reasons: (1) Multiple receivers send more ACKs for the retransmission; (2) Lost ACKs cause the sender to send duplicate packets. Thus, it is reasonably to predict that both the naive ARQ and accumulative ARQ perform worse when

the routing protocol supports alternative paths.

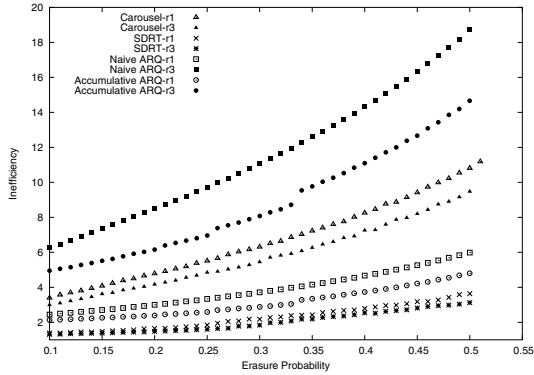


Fig. 3. Inefficiency vs. the erasure probability, where -r1 means the case of single receiver, and -r3 means the case of 3 receivers.

VI. RELATED WORK

In the literature, there are several seminal reliable transport protocols proposed for terrestrial sensor networks [18], [15], [9], [21]. However, due to the significant distinctions between underwater sensor networks and terrestrial sensor networks, these protocols are generally not suitable for underwater sensor networks. PSFQ (Pump Slowly and Fetch Quickly) [18], RMST (Reliable Multi-Segment Transport) [15] and RBC (Reliable Bursty Convergecast) [21], employ ARQ either in hop-by-hop transmission or end-to-end transfer. Thus they suffer from the problems inherited in ARQ when applied to underwater sensor networks. In [9], Kim et al. evaluate several methods of improving the reliability of data transport, namely erasure codes, retransmission and route fix. All the methods are implemented and evaluated in a real testbed of Mica2Dot. The results show that each of these methods are effective to overcome certain types of failures. The erasure codes evaluated in this work are Reed-Solomon codes [14]. In our paper, we have compared SDRT with Stop-and-Wait type of ARQ approaches, and our results also indicate that SDRT can help to support multi-path routing. Inspired by the work in [9], we believe that it is valuable to further investigate the design space in different underwater sensor network settings for various reliable data transfer methods.

VII. CONCLUSIONS

In this paper, we presented SDRT, a reliable data transport protocol for underwater sensor networks. SDRT segments data packets into blocks and encodes using SVT codes. The application of SVT codes increases channel utilization, reduces the number of packets transmitted in the network, and relieves both of the sender and receivers the burden of lost packet management. Therefore, SDRT is energy efficient and lightweight. Furthermore, the appropriate block size enables SDRT to effectively address the dynamic network topology problem.

Setting appropriate block size depends on the expected number of needed packets. We developed a model to estimate the number of packets actually needed for data recovery. We evaluated the accuracy of the model and the performance

of SDRT by simulations. The results show that our model accurately estimates the number of packets actually needed, and SDRT has much higher channel utilization and energy efficiency compared with other types of approaches.

REFERENCES

- [1] I. F. Akyildiz, D. Pompili, and T. Melodia, "Challenges for Efficient Communication in Underwater Acoustic Sensor Networks," *ACM SIGBED Review*, vol. Vol. 1, no. 1, July 2004.
- [2] W. Broecker and T.-H. Peng, "Tracers in the sea," *Eldigio Press, Lamont Doherty Earth Observatory of Columbia University, Palisades, NY*, p. 689, 1982.
- [3] J. W. Byers, J. Considine, M. Mitzenmacher, and S. Rost, "Informed Content Delivery Over Adaptive Overlay Networks," in *ACM SIGCOMM'02*, Pittsburgh, PA, August 2002.
- [4] J. W. Byers, M. Luby, and M. Mitzenmacher, "Accessing Multiple Mirror Sites in Parallel: Using Tornado Codes to Speed Up Downloads," in *INFOCOM'99*, New York, NY, March 1999.
- [5] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Approach to Reliable Distribution of Bulk Data," in *ACM SIGCOMM'98*, Vancouver, British Columbia, August 1998.
- [6] J.-H. Cui, J. Kong, M. Gerla, and S. Zhou, "Challenges: Building Scalable Mobile Underwater Wireless Sensor Networks for Aquatic Applications," *Special Issue of IEEE Network on Wireless Sensor Networking*, May 2006.
- [7] L. Freitag, M. Grund, S. Singh, J. Partan, P. Koski, and K. Ball, "The WHOI Micro-Modem: An Acoustic Communications and Navigation System for Multiple Platforms," in *IEEE Oceans Conference*, Washington DC, 2005.
- [8] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li., "Research Challenges and Applications for Underwater Sensor Networking," in *IEEE Wireless Communications and Networking Conference*, Las Vegas, Nevada, USA, April 2006.
- [9] S. Kim, R. Fonseca, and D. Culler, "Reliable Transfer on Wireless Sensor Networks," in *Frist IEEE International Conference on Sensor and Ad Hoc Communication and Networks (SECON'04)*, Santa Clara, CA, October 4-7 2004.
- [10] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical Loss-Resilient Codes," in *ACM STOC*, 1997, pp. 150–159.
- [11] J. Morris, "Optimal blocklengths for arq error control schemes," *IEEE Trans. Commun.*, vol. 27, pp. 488–493, Feb. 1979.
- [12] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 2000.
- [13] J. Proakis, E. Sozer, J. A. Rice, and M. Stojanovic, "Shallow Water Acoustic Networks," *IEEE Communications Magazines*, pp. 114–119, November 2001.
- [14] I. REED and G. Solomon, "Polynomial Codes over certain finite fields," in *Journal of the Society for Industrial and Applied Mathematics*, June 1960, pp. 8:300–304.
- [15] F. Stann and J. Heidemann, "RMST: Reliable Data Transport in Sensor Networks," in *First IEEE International Workshop on Sensor Net Protocols and Applications(SNPA)*, Anchorage, Alaska, USA, May 2003.
- [16] M. Stojanovic, "Optimization of a Data Link Protocol for an Underwater Acoustic Channel," in *IEEE OCEANS'05 Conference*, Brest, France, June 2005.
- [17] P. Turney, "An improved stop-and-wait arq logic for data transmission in mobile radio systems," *IEEE Trans. Commun.*, vol. 29, pp. 68–71, Jan 1981.
- [18] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy, "PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks," in *WSNA'02*, Atlanta, Georgia, USA, September 2002.
- [19] G. G. Xie and J. Gibson, "A Networking Protocol for Underwater Acoustic Networks," in *Technical Report TR-CS-00-02*, Department of Computer Science, Naval Postgraduate School, December 2000.
- [20] P. Xie and J.-H. Cui, "SDRT: A Reliable Data Transport Protocol for Underwater Sensor Networks," *UCONN CSE Technical Report: UbiNet-TR06-03 (BECAT/CSE-TR-06-14)*, February 2006.
- [21] H. Zhang, A. Arora, Y. Choi, and M. G. Gouda, "Reliable Bursty Convergecast in Wireless Sensor Networks," in *Mobihoc'05*, Urbana-Champaign, Illinois, USA, May 25-27 2005.