Reliable and Efficient Data Transfer for Underwater Acoustic Networks

Ph.D Prospectus
by
Haining Mo

Department of Computer Science and Engineering
University of Connecticut

Abstract. As an emerging research area, Underwater Acoustic Networks (UANs) have attracted tremendous interests in last several years. Reliable and efficient data transfer is of critical importance for UANs since it serves as the essential underlying service for various tasks. Due to the unique features of UANs including the long propagation delay, low bandwidth and high error probability, reliable and efficient data transfer has been facing great challenges. In this dissertation work, we plan to tackle this problem from three different perspectives.

First, we design a Coding based multi-hop Coordinated Reliable Data Transfer (CCRDT) protocol for multi-hop string topology UANs. GF(256) Random Linear Coding is employed to enhance reliability and efficiency over one hop. A multi-hop coordination scheme is proposed to eliminate collisions and improve throughput. CCRDT has been implemented on real UAN nodes and extensive lab tests has been conducted to show its advantages over existing approaches.

Second, we propose a Selective ARQ and Slotted Handshake based Access (SASHA) for arbitrary topology UANs. SASHA embraces the most commonly employed techniques in coordination based UAN MAC design. We implemented SASHA on real UAN nodes and conducted a sea test to evaluate its performance. We investigated the hop-by-hop and end-to-end behavior of SASHA. From the experimental data, important issues have been discovered and corresponding design guidelines are suggested.

Last, to address the reliable and efficient broadcast problem for UANs, we propose a Two-phase Broadcast (TBS) scheme. TBS does not rely on topology or neighbor information and thus is more adaptive to the dynamic changes in UANs. TBS includes two phases: the Fast Spreading phase and the Data Recovery phase. The Fast Spreading phase combines opportunistic overhearing and network coding to improve broadcast efficiency. The Data Recovery phase aims to guarantee reliability and reduce interference probability.

In this prospectus, we will present these three works in detail. We will also discuss the future directions to complete this Ph.D. dissertation.

1 Introduction

The last decade has witnessed a remarkable progress in Underwater Acoustic Networks (UANs) because of its various applications including underwater environment monitoring and tactical surveillance [1–4]. In UANs, acoustic channels are usually employed for signal transmission in the water [5–7]. The data propagation delay is long due to the low propagation speed of acoustic signals (around 1500m/s). Additionally, UAN channels have very low data rates because of the absorption, multi-path and fading. Moreover, UANs usually suffer from high error probabilities because of error prone underwater acoustic channels as well as the intermittent network connectivity.

The unique features of UANs impose great challenges to data transfer in underwater environments [8–11] and conventional technologies for Terrestrial Wireless Sensor Networks (TWSNs) cannot be directly applied. In this dissertation work, we propose three schemes to tackle the reliable and efficient data transfer problem for UANs from three different perspectives.

- Reliable and efficient data transfer for multi-hop string topology networks is always a desirable feature for UANs. From observations in field tests, we point out critical issues with existing approaches. Motivated by the discovered issues, we design and implement a Coding based multi-hop Coordinated Reliable Data Transfer (CCRDT) protocol. CCRDT combines GF(256) Random Linear Coding and selective repeat to
guarantee the per-hop reliability. A multi-hop coordination scheme is also proposed to enable transmission pipelining and eliminate packet collisions. We implemented CCRDT on a lab testbed and conducted extensive tests to evaluate its performance. Preliminary results show that CCRDT achieves a higher end-to-end throughput than other existing reliable data transfer protocols which have been implemented in real systems.

- CCRDT is tailored for string topology UANs and is essentially a scheduling based Medium Access Control (MAC) protocol. To implement a reliable and efficient data transfer scheme for UANs with arbitrary topologies, we propose SASHA, which is Selective ARQ and Slotted Handshake based Access. SASHA embraces most essential techniques in the design of coordination based UAN MAC protocols, including time slotting, handshake, collision avoidance and selective ARQ. More importantly, we implemented SASHA on real UAN nodes and conducted a sea test in Atlantic ocean to evaluate its performance. With the experimental data, we are able to study how the aforementioned techniques affect the performance of SASHA. We also analyze the hop-by-hop and end-to-end behavior of SASHA. From the findings, some issues are discovered and the corresponding design guidelines are emerged.

- CCRDT and SASHA are designed for reliable unicast and multicast in UANs. On the other hand, reliable and efficient broadcast is also of critical importance for UANs. To address this problem, we propose a Two-phase Broadcast Scheme (TBS) for UANs. TBS includes two phases: Fast Spreading phase and Data Recovery phase. It does not require topology or neighbor information. The Fast Spreading phase combines opportunistic overhearing and network coding to improve broadcast efficiency. The Data Recovery phase is to guarantee reliability and minimize interference probability. Preliminary results show that TBS achieves a smaller broadcast completion time and a comparable energy efficiency.

2 Coding based multi-hop Coordinated Reliable Data Transfer

2.1 Motivation

Conventional end-to-end and hop-by-hop reliable data transfer schemes for TWSNs tend to be ineffective for UANs because of the high retransmission probability caused by error-prone channels. Every retransmission leads to significant growth in end-to-end delay because of the large propagation delays in the underwater environment [12]. To address this issue, a couple of approaches dedicated to UAN data transfer have been proposed [13, 14]. Among them, we have chosen to implement SDRT [15] and Aqua-SARQ (a TCP-like end-to-end approach based on sliding window and selective repeat) on real UAN nodes. Additionally, we have performed tests for both protocols in the lab as well as in a lake environment [6] to evaluate their performance. Based on our real-world tests and evaluations, we discovered some practical issues with these two representative protocols.

For SDRT, it is difficult to implement the coding ratio estimation scheme on real UAN nodes. The underlying reason is that SDRT employs SVT coding, which is a sparse coding scheme. SVT coding can make sure encoded packets are able to recover K data packets only when K is large enough. However, in real world UAN applications, K is usually smaller than 100. This means that even without any packet loss, K encoded packets sent by the sender cannot recover K data packets at the receiver. As a result, the optimal coding redundancy of SDRT K’ depends on the Packet Error Rate (PER), and also on the properties of SVT coding, including the coding degree and block size. To address this problem, we employ a GF(256) Random Linear Coding (RLC), which can almost 100% guarantee that K encoded packets are able to recover K data packets [16]. Using RLC, the coding ratio estimation becomes simple since it only depends on the PER.

The second issue we discovered with SDRT is that it lacks a multi-hop coordination scheme and consequently collisions occur frequently in multi-hop UANs. Three types of collisions were observed: Data-ACK, Sending-receiving and Overhearing collisions. These collisions have been well studied in both TWSNs and UANs. They are alleviated by either RTS/CTS based schemes [17] or scheduling based methods [18]. Considering the non-trivial overhead of RTS/CTS imposed by long propagation delays as well as the characteristics of a string topology network, we adopt a scheduling based multi-hop coordination scheme to eliminate collisions.

We have also implemented Aqua-SARQ, which is an end-to-end approach that depends on sliding window and selective repeat to ensure reliability. Obviously, it lacks a pipelining scheme to allow multiple senders at one time. Therefore, we also incorporate pipelining into our multi-hop coordination scheme to further improve end-to-end throughput.
Therefore, in order to address the aforementioned issues and to develop a practical reliable data transfer protocol that can work in real world UANs, we have designed and implemented CCRDT, a Coding based multi-hop Coordinated Reliable Data Transfer protocol for UANs. For the per-hop data transfer, CCRDT employs a GF(256) Random Linear Coding scheme and selective repeat to ensure reliability and efficiency. The chosen coding scheme makes coding ratio estimation much easier to implement on real UAN nodes. For the multi-hop network, CCRDT utilizes a multi-hop coordination scheme to eliminate collisions. This multi-hop coordination also implements a pipelining scheme that allows multiple nodes to transmit simultaneously and therefore largely increases end-to-end throughput.

2.2 Protocol Design

**GF(256) Random Linear Coding** We use $X_1, X_2, ..., X_k$ to denote $K$ data packets in one block. The $K$ data packets are linearly combined to obtain $K'$ encoded packets, denoted as $Y_1, Y_2, ..., Y_{K'}$, where $Y_i = \sum_{j=1}^{K} a_{ij} X_j$. Here $a_{ij}$ is the encoding coefficient randomly picked from a Galois Field GF$(2^q)$. $(a_{i1}, a_{i2}, ..., a_{ik})$ is defined as the encoding vector for $Y_i$. $\frac{K'}{K}$ is usually defined as the coding ratio, denoted by $r$.

The success probability of Random Linear Coding to recover $K$ data packets from $K'$ encoded packets depends on the value of $q$, namely the number of elements in the Galois Field. In other words, with a large enough $q$, we can always guarantee a high decoding success probability. In our tests, with GF(256) Random Linear Coding, the probability that $K$ encoded packets fail to recover $K$ data packets is negligible.

We choose GF(256) Random Linear Coding due to its strong data recovery capability, which significantly simplifies the coding ratio estimation. Its decoding complexity is $O(K^3)$, where $K$ is the number of encoded packets. Although the decoding complexity is larger than Binary Random Linear Coding and other sparse coding schemes, the computationally powerful UAN nodes and a relatively small $K$ in real world UAN applications guarantee that the decoding can be completed in real time.

**Per-hop Data Transfer** The per-hop data transfer of CCRDT is designed to guarantee the reliable delivery of one block of data packets over one hop with minimal delay. It is closely coupled with the GF(256) Random Linear Coding. When the sender gets the chance to send out a block of $K$ data packets, it will generate $K' = rK$ encoded packets using GF(256) Random Linear Coding where $r$ is defined as its current coding ratio. After that, the sender will send the $K'$ encoded packets to the receiver and some packets may get lost due to the channel erasure. If the receiver is able to recover the $K$ data packets, it will send back an ACK. Otherwise, it will send back a NACK indicating a data recovery failure. Both the ACK and NACK include the information about how many packets are received at the receiver in the last transmission and therefore the sender can update its PER information and in turn update its coding ratio $r$ whenever it receives an ACK or NACK. Upon receiving a NACK, the sender will send out more encoded packets based on the updated $r$ and the number of encoded packets still needed at the receiver. This procedure will continue until an ACK is received at the sender. Using the selective repeat, the reliability over one hop can be achieved.

With GF(256) Random Linear Coding, setting the coding ratio $r$ becomes simple. Whenever the sender receives an ACK or NACK, it updates its PER information on this hop and then sets $r$ to be $1 - \frac{1}{1 - PER}$. Then, if the sender sends out $\frac{K}{1 - PER}$ encoded packets, $K$ encoded data packets can be received at the receiver with a high probability. Due to the power of GF(256) Random Linear Coding, the $K$ data packets can be recovered. In this way, we can guarantee with a high probability that one transmission of $\frac{K}{1 - PER}$ encoded packets is enough to recover $K$ data packets and therefore the data transfer delay on one hop can be minimized.

**Multi-hop Coordination** In Section 2.2, we discussed a node’s behavior after it gets a chance to send out a block of data packets. In this section, we will discuss when a node in a multi-hop UAN should be assigned a chance to send, which is determined by the multi-hop coordination scheme of CCRDT. The target of multi-hop coordination is to realize pipelining and to eliminate collisions.

In UANs, to allow multiple nodes to send out simultaneously without causing collisions, the senders need to be at least two hops away from each other. Inspired by this observation, we introduce the notion of Collision Avoidance ID (CAID) and assign the same CAID to a group of nodes two hops away from each other. Since the nodes with the same CAID are two hops away, we actually only need three different CAIDs (2, 1, 0) to cover all the nodes. We also propose a source-initiated distributed way to assign CAIDs to all the nodes in a multi-hop
UAN with the string topology. The source node starts with CAID set to 2. Every node sends its CAID to its downstream node. A node, upon receiving the CAID from its upstream node, decreases the received CAID by 1 and uses this value as its own CAID. If a node receives a CAID equal to 0, it sets its own CAID to 2. This process continues until the sink node sets its CAID.

Based on the idea that nodes with the same CAID can send out simultaneously, we design a time slot based multi-hop coordination scheme for CCRDT, which is shown in Fig. 1. In the first three time slots, since node 4, 5 and 6 have no data block to send out, only one node is sending at one time slot. Starting from the 4th time slot, there are two nodes with the same CAID sending at one time slot. For instance, in the 4th time slot, node 4 is sending block $j - 1$ while node 1 is sending block $j$. If we increase the number of nodes in the network, there can be 3 or more nodes sending out simultaneously at one time slot. In this way, in a network with 7 nodes as shown in Fig. 1, starting from the 6th time slot, there will be a new block delivered to the sink node every 3 times slots. Without the multi-hop coordination, a new block can be delivered to the sink node every 6 time slots. Therefore, using the multi-hop coordination scheme, the end-to-end throughput can be significantly improved and no collision is incurred.

Obviously, the efficiency of the multi-hop coordination heavily depends on the time slot length $T_S$, which is the same for every node. If $T_S$ is too small, a block of data packets cannot be recovered at the receiver within one time slot. Therefore in the next 2 time slots, the 2 succeeding nodes have nothing to send out. This will waste in total 3 time slots. On the other hand, if $T_S$ is too large, after the data block is recovered at the receiver and the ACK is received at the sender, the remaining time in one time slot is also going to be wasted.

As discussed in Section 2.2, with coding ratio estimation and GF(256) Random Linear Coding, we guarantee with a high probability that one transmission of encoded packets is enough for the receiver to recover a block of data packets. Therefore, we set $T_S$ to be large enough to cover the transfer of a block of encoded packets and an ACK from the receiver to the sender. The transfer delay of the encoded packets and the ACK includes both the transmission delay and the propagation delay.

$$T_S = \frac{r \cdot K \cdot L \cdot 8}{R} + \frac{D \cdot 2}{V} + \frac{1 \cdot 8}{R} = \frac{8(rKL + 1)}{R} + \frac{2D}{V}$$

Here $r$ is the coding ratio; $K$ is the data block size; $L$ is the packet length; $R$ is the modem acoustic bit rate; $D$ is the distance of one hop; $V$ is the sound speed in the water and an ACK packet has only one byte (we ignore the packet header for simplicity).

The coding ratio $r$ is decided by PER. Since $T_S$ is the same for every node, we are going to decide $T_S$ by the highest PER in the network. However, since we consider a homogeneous string topology network, the PERs on all the hops are close to each other. Because we have no idea about the PER information within the network in the beginning, in the current implementation, we need to pre-run a program to collect the PER information over multiple hops to set $T_S$. In the future, we plan to develop a more adaptive scheme to dynamically adjust $T_S$ based on the PER information in the network.
Therefore, using the multi-hop coordination, a block of data packets can be recovered within one time slot. Multiple nodes can send simultaneously at one time slot without causing collisions. In this way, the end-to-end throughput can be largely improved.

2.3 Protocol Implementation

To better verify the design and evaluate the performance, we implemented CCRDT on real UAN nodes. In our system, a UAN node is composed of a Gumstix [19] working as the controller for the UAN node and a Teledyne Benthos Modem [20] conducting the acoustic communications to send and receive packets.

The software suite on a UAN node include Embedded Linux and an underwater network protocol stack: Aqua-NET [21]. Aqua-NET is a layered protocol stack including the physical layer, data link layer, MAC layer, routing layer, transport layer and application layer. We implemented CCRDT on the data link layer. For the other layers, we use Poisson Traffic Generator on the application layer, which generates traffic with time interval between every two packets following a Poisson distribution, a generic transport layer, Static Routing on the routing layer, Broadcast MAC on the MAC layer, and a Benthos modem driver on the physical layer. The overall architecture of the CCRDT implementation is shown in Fig. 2.

2.4 Preliminary Experiment Results

We built a string topology network with 3 types of Teledyne Benthos modems: ATM-920 modem, UDB-9000 Deck Box modem and SM-75 Smart modem [20]. The network parameters are as follows unless otherwise specified: the hop count $N$ is 4; the block size $K$ is 5, meaning every 5 data packets are grouped into one block and encoded; the packet length $L$ is 200; the modem acoustic bit rate is 800bps; the sliding window size for Aqua-SARQ is 1, which we will discuss later.

**Impact of Packet Length** We investigate the impact of the packet length on the end-to-end throughput of the three protocols. The result is shown in Fig. 3(a). With different packet lengths from 50 to 800, CCRDT achieves a much better end-to-end throughput than SDRT or Aqua-SARQ. One reason is that CCRDT utilizes coding ratio estimation and GF(256) Random Linear Coding to make sure that one transmission of encoded packets is enough. This can be proved by the CCRDT log file, which shows that no retransmission is incurred during the experiment. By contrast, both SDRT and Aqua-SARQ suffer from retransmissions degrading system throughput due to the packet loss. Another reason is that CCRDT enables a pipelining scheme which allows node 1 and node 4 to send simultaneously in one time slot (in a 4-hop string topology). Compared with the end-to-end Aqua-SARQ, the pipelining leads to a larger throughput. The third reason is that the multi-hop coordination of CCRDT completely eliminates the collisions within the network. Unlike SDRT, CCRDT does not suffer from any Data-ACK, Sending-Receiving or Overhearing collision.

Also, we observe that as the packet length grows, the end-to-end throughput of all three protocols grows larger, which makes sense since more bits can be delivered at one time. However, we do notice that the throughput of CCRDT grows faster than SDRT and Aqua-SARQ with larger packet lengths. The reason is that as the packet length becomes larger, so does the PER on one hop. Since CCRDT adapts to the increasing PERs better than SDRT and Aqua-SARQ due to the coding ratio estimation, it suffers less from a larger PER.

**Impact of Modem Acoustic Bit Rate** The impact of the modem acoustic bit rate on the end-to-end throughput is shown in Fig. 3(b). Here the block size is 5; hop count is 4 and packet length is 200. Again CCRDT outperforms SDRT and Aqua-SARQ because of the GF(256) Random Linear Coding, the coding ratio estimation and the multi-hop coordination.

**Impact of Hop Count** In this section, we study how hop count affects the end-to-end throughput of the three protocols. The result is shown in Fig. 3(c). We can see that CCRDT achieves the highest throughput among the three protocols. Also as the hop count grows, the throughput of all three protocols decreases.

Another fact we observed is that when the hop count is small, SDRT has a better throughput than Aqua-SARQ. However, as hop count grows larger, the throughput of SDRT degrades faster than the other two
protocols. This is because with more hops, SDRT has a larger chance to suffer from collisions. In Aqua-SARQ, we set the sliding window size to be 1. It seems that a smaller window size will lead to a lower throughput since only one packet can be delivered to the sink node at a time. However, setting sliding window size to be 1 actually eliminates the Sending-Receiving collisions and achieves a higher throughput. This is why Aqua-SARQ has a better performance than SDRT with a larger hop count. This analysis can be verified by our experiment on how the sliding window size affects the Aqua-SARQ throughput, as shown in Fig. 4. We can see that a sliding window with size 2 achieves a bit higher throughput than a sliding window with size set to 1. However, after that, the end-to-end throughput decreases as the sliding window grows larger due to the Sending-Receiving collisions. In our experiment, due to the simplicity and the slight difference, we still set sliding window size to be 1 rather than 2 for Aqua-SARQ.

Overhead In this section, we study the overhead incurred by different protocols. Due to the time and space limitation, we only compare the overhead of CCRDT and Aqua-SARQ here, as shown in Fig. 5. The extra packets in CCRDT include the ACK packet at each hop and the unnecessary encoded packets sent out. The extra packets in Aqua-SARQ are the end-to-end ACKs. In this experiment, block size \( K \) is set to 5 and packet length \( L \) is set to 200. We vary the hop count from 1 to 5.

We can see that the overhead of Aqua-SARQ decreases with an increasing hop count. The reason is that with 1 hop, every data packet transmitted leads to an ACK while with 5 hops, 5 transmitted data packets (one data packet going through 5 hops) incurs one ACK. As to CCRDT, the overhead stays mostly unchanged with different hop counts. For one reason, CCRDT uses hop-by-hop ACK and therefore each block of data packets causes one ACK on every hop. For another, the coding ratio estimation and the GF(256) Random Linear Coding minimizes the amount of unnecessary encoded packets. Also we can see that with a smaller hop count (less than 5), CCRDT achieves a smaller overhead than Aqua-SARQ.

2.5 Summary

In this work, motivated by the issues observed with existing protocols implemented in real systems, we propose CCRDT: a Coding based multi-hop Coordinated Reliable Data Transfer protocol. For the per-hop data transfer, CCRDT employs a GF(256) Random Linear Coding due to its strong data recovery capability when data block size is small. A coding ratio estimation scheme is proposed to ensure the efficiency of data transfer in one hop. A multi-hop coordination scheme is designed not only to allow pipelining but also to avoid collisions. CCRDT has been implemented in real systems. Preliminary experimental tests show that CCRDT outperforms two protocols implemented and tested in the past.

3 Selective ARQ and Slotted Handshake based Access

3.1 Motivation

UAN MAC protocols can be coarsely classified into two categories: random access based ones and coordination based ones [22]. Random access based protocols introduce a minimum overhead from control packets but usually
suffer more from collisions. By contrast, coordination based protocols incur a larger overhead but usually achieve a much smaller collision probability. In a UAN with a relatively high traffic load and node density, a random access based protocol can severely degrade the overall efficiency of data transmission due to the frequent occurrence of collisions and therefore we focus on coordination based MAC protocols in this work.

Some techniques are commonly employed by coordination based MAC protocols, including handshake, time slotting and collision avoidance [17, 23, 24]. Great efforts have been devoted to simulation based analysis of these popular techniques. However, simulations have two limitations. On one hand, most simulators have limited capability in reflecting the highly dynamic nature of the underwater environment and therefore there could be a nontrivial gap between the performance of a protocol in a simulator and that in real world sea tests [25]. On the other hand, there may be some facts unknown to the simulators that can only be revealed in real world sea tests. For instance, Pu et al. in [26] discovered that the long preamble of acoustic modems could significantly degrade the performance of UAN MAC protocols, which was not taken into consideration by simulators before.

Based on the above discussion, we understand that designing a MAC protocol involving the aforementioned techniques as well as implementing and testing it in real world UANs will be very helpful to study how general coordination based MAC protocols perform in real world. To this end, we designed a MAC protocol called Selective ARQ and Slotted Handshake based Access (SASHA) [27]. As suggested by the name, SASHA is a coordination based protocol utilizing selective ARQ, time slotting, handshake and collision avoidance. Besides the design, we implemented SASHA on UAN nodes by utilizing Gumstix [19], Teledyne Benthos Modem [20] and a UAN protocol stack Aqua-NET [21]. Beyond that, we conducted a sea test to evaluate the performance of SASHA. A 9-node UAN was deployed 120 km off New Jersey shore between September 6th and 10th, 2012.

3.2 Protocol Description

**SASHA Overview** The overall work flow of SASHA is illustrated in Fig. 6. Node $i$ and $i+1$ are the sender and the receiver while node $i-1$ and $i+2$ are two bystanders that can overhear packet transmissions between node $i$ and $i+1$. An RTS/CTS exchange lasting two time slots is initiated between node $i$ and $i+1$ to establish a conversation. Node $i-1$ which overhears the RTS and node $i+2$ which overhears the CTS, will back-off. After that, in the beginning of the next time slot, node $i$ sends out an HDR, followed by a DATA packet train of 3 packets, in this example. HDR also causes node $i-1$ to keep silent. Due to the channel erasure, only 2 DATA packets are received at node $i+1$. Therefore node $i+1$ sends out a NACK, causing node $i+2$ to keep silent. Upon receiving the NACK, node $i$ sends out an HDR in the next time slot, followed by the retransmitted DATA packet 2. This retransmission continues until an ACK is received at node $i$. Note that the transmissions of the control packets and the first DATA packet in the packet train are initiated in the beginning of a time slot.
Besides DATA packets, there are 5 types of control packets in SASHA. RTS/CTS are used for the handshake procedure. HDR is a new type of control packet sent out prior to the transmission/retransmission of DATA packets. It carries the information on how many DATA packets will be sent out following the transmission of HDR. HDR servers the purpose of both selective ARQ and collision avoidance. First, it informs the receiver of the expected number of DATA packets and therefore the receiver is able to construct a NACK/ACK packet. Second, a bystander overhearing an HDR can estimate the duration of the coming data transmission session based on the information embedded in HDR. Therefore it is able to choose an appropriate back-off period. NACK and ACK packets are used for selective ARQ [28]. Although HDR seems to be enough for the purpose of collision avoidance, we let RTS/CTS/NACK carry similar information to HDR, which is the number of DATA packets in the coming transmission session. A bystander overhearing an RTS/CTS/NACK can thus back-off accordingly. The purpose of this decision is to add redundancy and therefore an HDR loss will not invalidate collision avoidance.

The state machine of SASHA is shown in Fig. 7. It is composed of two main threads: the sending thread and the receiving thread, which respectively reflects state transitions of the sending and receiving procedure.

Fig. 6. SASHA overall work flow

Fig. 7. SASHA state machine

**Selective ARQ** The handshake, time slotting and collision avoidance scheme in SASHA is identical to [17]. In addition to these schemes, we find that adding selective ARQ to coordination based UAN MAC protocols is essential.

In coordination based MAC protocols, a nontrivial overhead is incurred by the handshake procedure. Therefore, to improve channel utilization and energy efficiency, a packet train of multiple DATA packets is usually transmitted after a successful handshake. Most current implementations of UAN MAC protocols have not incorporated selective ARQ into this packet train scheme. After a successful handshake, only one DATA transmission session is allowed. This means that lost/unacked DATA packets will not be retransmitted immediately. Instead, the communication pair has to re-compete for the channel via a new handshake procedure in order to initiate a retransmission. With this scheme, an ACK loss will lead to not only a retransmission but also a re-competition since the sender would assume that no DATA packet was received. If ACK loss happens frequently, a significant overhead from handshake will be imposed. We may assume that the probability of ACK loss is much smaller than that of DATA packets since an ACK is much shorter. However, channel asymmetry has been observed in [22], which discovered that in real world UANs, on a single channel, the backward link may have a much worse link quality than the forward link or vice versa.
Motivated by the above discussion, SASHA implemented a selective ARQ scheme associated with the packet train mechanism. After a successful handshake, multiple consecutive DATA retransmissions are allowed until an ACK is received acknowledging all the DATA packets in the packet train. If a NACK is received, the sender will retransmit the lost DATA packets. If no NACK/ACK is received after time out, the sender will retransmit all the last transmitted DATA packets. Therefore with selective ARQ, one handshake can guarantee that the whole packet train can be received successfully. Selective ARQ effectively tackles the ACK loss issue and reduces the overhead brought by handshake.

3.3 Sea Test Setting

To study the behavior and performance of SASHA, we conducted a sea test between September 6th and 10th, at Atlantic Ocean, 120 km off New Jersey shore. During the sea test, we deployed 9 UAN nodes, at the locations as shown in Fig. 8. The nodes were located 120 km offshore at the depth around 80 m. The average height of the sea wave during the test was between 1.5 m and 2.5 m. The deployment area embraced wave, tide, salinity and temperature variance as well as fish and marine mammal movement. The average distance between two adjacent UAN nodes is 1 km while the distance between two end nodes is 7.3 km.

We managed to carry out 3 successful tests. The modem power level, modem operation rates and node counts of different tests are shown in Table 1.

<table>
<thead>
<tr>
<th>Test No.</th>
<th>Power Level</th>
<th>Operation Rate</th>
<th>Node Count</th>
<th>Packet Train Len</th>
<th>Traffic Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>300bps</td>
<td>5</td>
<td>100B</td>
<td>0.015</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>600bps</td>
<td>8</td>
<td>200B</td>
<td>0.005</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>300bps</td>
<td>9</td>
<td>200B</td>
<td>0.005</td>
</tr>
</tbody>
</table>

We formed 5-node, 8-node and 9-node (correspondingly 4-hop, 7-hop and 8-hop) networks. In all the three tests, we chose to set the transmission powers of Benthos modems to the lowest level in order to ensure that a node could reach only its immediate neighbors. The packet length and packet train length were both relatively small, considering the high probability of packet loss. The traffic generation rates were also selected to be low. Due to the significantly long end-to-end delays we observed during the tests, a high traffic generation rate could easily overwhelm the network.
3.4 Preliminary Sea Test Results

We conducted sea tests to study the behavior and performance of SASHA in real world UANs. We are mostly interested into how SASHA behaves hop-by-hop wise. This mainly includes the packet delivery delay on a single hop and what factors contribute to this delay. The hop-by-hop behavior of SASHA fundamentally leads to its end-to-end performance. It provides insights on how the techniques employed by SASHA perform in real world applications.

In SASHA, a node essentially has three types of actions during its lifetime. If it has nothing queued in its incoming queue, it simply stays in the IDLE state. Otherwise, the node takes one of the two actions. If the node is informed of a potential collision or fails to complete a handshake due to RTS/CTS loss, the node backs-off. In this case, the packets are queued at the node. We call the delay related to this type of action queuing delay. On the other hand, if the node senses no conflicting activity and succeeds in completing a handshake, a packet train will be sent out. The delay before the entire packet train is successfully received at the receiver is called transmission delay. Next we will analyze the hop-by-hop transmission delay and queuing delay in detail.

Transmission Delay on One Hop

The average transmission delays of the 4-hop, 7-hop and 8-hop tests are shown in Fig. 9. Transmission delay generally consists of the transmission time and propagation time of RTS, CTS, HDR, DATA, ACK and possibly NACK and retransmitted DATA. As shown in Fig. 9(a), the transmission delays over different hops in the 4-hop test were pretty consistent. The reason is that during this test, DATA packet loss rarely happened and therefore few retransmissions were involved. Nevertheless, the small variance on different hops mainly originated from the propagation delay difference on different hops, which was caused by distance difference as well as temperature and salinity variance affecting the sound propagation speed.

Similarly, in Fig. 9(b) and 9(c), transmission delays also maintained relatively stable values within the network expect that there was a peak point in both the 7-hop and 8-hop tests. From our sea test log file, we found that the much more significant transmission delay on Hop 5 in the 7-hop test, for instance, came from the retransmissions of DATA packets. On that very link, DATA packet loss occurred a lot and triggered the selective ARQ procedure and therefore a larger transmission delay was observed. Admittedly, selective ARQ increases the transmission delay for a packet train. However, it allows the entire packet train to be received with only one channel competition. Without selective ARQ, multiple competitions may be required to deliver the packet train, as discussed in Section 3.2. Due to the overhead from the handshake procedure, the overall per-hop delivery delay for the packet train would be significantly larger.

Therefore, the factor affecting transmission delay is the link quality of a given hop. On a hop with a poor link quality, the loss of DATA, NACK and ACK packets leads to retransmissions and therefore increases the transmission delay.

Queuing Delay on One Hop

Compared with the per-hop transmission delay, the queuing delay on one hop was much larger and accounted as the major part of the delivery delay, as shown in Fig. 10. Queuing delay of a DATA packet is defined to be the time from when the packet is received at a node to when the last RTS is sent
out leading to a successful handshake. Earlier we mentioned that the queuing of a DATA packet is because of the back-off due to overhearing ongoing transmissions in the neighborhood or the failure of handshake incurred by RTS/CTS loss. During the tests, we discovered that another factor significantly contributing to the large queuing delay was the transmission range uncertainty which caused unexpected collisions.

In the 4-hop test, the largest queuing delay appeared on Hop 1. The reason is that we employed 0.015 as the traffic generation rate, which was the highest among all three tests. A traffic generation rate of 0.015 means that a DATA packet was generated at the source node averagely every 66 seconds. As we can see from Fig. 10(a), the average per-hop delivery delay in the 4-hop test was more than 150 seconds. Obviously, 0.015 was too aggressive. Under this circumstance, a lot of DATA packets were queued at the source node after being generated. This is why the first hop experienced the largest queuing and delivery delay. Another consequence of the high traffic generation rate was the increasing channel competition, which lowered the handshake success rate and therefore led to a large queuing delay. The underlying reason is that with a higher traffic generation rate, the source node tended to send out RTS at a higher frequency, which translated into higher RTS sending rates on succeeding nodes as well.

For the 7-hop test, the three middle hops, namely Hop 3, 4 and 5 experienced larger queuing as well as delivery delays than the other hops. The reason is that the middle nodes had more immediate neighbors than the edge nodes in the string topology. Therefore, they were likely to overhear more RTS/CTS from neighbors in both directions, which incurred larger back-off periods. Hop 7 had the lowest queuing delay because of two reasons. First, the traffic rate at the end of the network was much lower than those on the first several hops. Second, the sink node had only one immediate neighbor, which suffered from less competitions than the middle nodes. In this test, the traffic generation rate was 0.005, three times lower than that in the 4-hop test. This explains why the 7-hop test had overall lower queuing delays than the 4-hop test.

In the 8-hop test, we found an abnormally large queuing delay on hop 5, as shown in Fig. 10(c). There are already two known factors leading to this phenomenon. On one hand, the ongoing transmissions in the neighborhood caused both the sender and the receiver to back-off. On the other hand, Hop 5 suffered from a higher packet loss rate.

However, after checking the test log files, we observed an extraordinary amount of handshake failures on Hop 5 due to RTS/CTS loss. This could not be explained by channel erasure only since RTS/CTS loss happened at a much higher rate than other control or DATA packets. After examination, we found the reason is that the two nodes on Hop 8 were able to reach the receiver on Hop 5. As a consequence, simultaneous transmissions of RTS on Hop 5 and 8 would cause collisions at Hop 5, which imposed back-offs. In another word, Hop 5 suffered from competitions with its immediate neighbors as well as with neighbors that were two hops away. With the frequent occurrence of this type of extra collisions, the queuing delay on Hop 5 became extremely long.

SASHA did not expect the above type of collision. The root reason is that even with very carefully selected modem transmission power levels, we still could not guarantee that a node was able and only able to communicate with its two immediate neighbors. This phenomenon is defined as transmission range uncertainty. In the above example, nodes on Hop 8 were able to talk to nodes on Hop 5. To address this problem, one feasible solution might be to obtain the accurate network topology rather than assuming a network topology. To this end, a dynamic topology probing approach has to be in place.

Fig. 10. Queuing and delivery delays of 4-hop, 7-hop and 8-hop networks in SASHA
End-to-end Performance. The overall end-to-end delivery delay of the 4-hop, 7-hop and 8-hop tests are shown in Fig. 11. As the network size grows larger, so does the end-to-end delivery delay. The significant growth in the delay for the 8-hop test stemmed from the unexpected collisions caused by transmission range uncertainty. The transmission range uncertainty problem becomes severer in larger networks, where more nodes lead to more overhearing and a larger collision probability.

The end-to-end throughput decreases with the increase of the network size, as shown in Fig. 12. Also we can see that the achieved throughput in the 7-hop and 8-hop UANs were actually pretty low, which proved the difficulty of networked communication in real world underwater environments. In terms of end-to-end delivery ratio, all three tests achieved 100% delivery ratio. Since SASHA incorporates selective ARQ, it can guarantee the end-to-end reliability of each DATA packet.

3.5 Summary

In this work, towards gaining a better understanding on how UAN MAC protocols perform in real world underwater environments, we implemented SASHA on UAN nodes. SASHA employs some essential techniques in the design of coordination based UAN MAC protocols, including selective ARQ, time slotting, handshake and collision avoidance. In addition, a sea test was conducted at Atlantic Ocean to test SASHA. We analyze the performance of SASHA both hop-by-hop wise and end-to-end wise. Particularly, we investigate the transmission delay and queuing delay of a data packet on a single hop and what factors affect these two delays. Through investigation, we give some design guidelines to improve the performance of MAC protocols in real world UANs.

4 Two-phase Broadcast Scheme for Underwater Acoustic Networks

4.1 Motivation

Reliable broadcast has always been a desirable feature for wireless networks. Applications such as updating the firmware/operating system on every node in the network and distributing a critical data file network-wide are common and required by various tasks in both TWSNs and UANs.

Broadcast in TWSNs Various approaches have been proposed for broadcast in Ad-Hoc Networks, with blind flooding being the simplest one [29]. Blind flooding achieves the best reliability at the cost of the broadcast storm [30]. To alleviate the broadcast storm problem, extensive research efforts have been devoted to the selection of an optimal set of rebroadcasting nodes. The proposed mechanisms can be coarsely categorized into probability based methods, area based methods, neighbor knowledge based methods and cluster based method [31].
However, the aforementioned mechanisms tend to be impractical and inefficient for UANs considering the special features of the underwater environment. In pure probability based methods, a node rebroadcasts with either a static or an adaptive probability, which is inversely proportional to the number of neighbors, for example [32]. Static probability is inefficient because the highly dynamic nature of UANs imposes great challenges to predetermine the rebroadcast probability. Adaptive probability is not a good option for UANs either due to the difficulty to obtain and maintain the neighbor information.

Neighbor knowledge based methods [33, 34] are inefficient for UANs due to two reasons. On the one hand, the large propagation delay in UANs and the long preamble of underwater acoustic modems [26] incur a significant overhead to message exchanges. According to [26], Benthos modem [20] imposes a preamble lasting 1.5 seconds. This implies that for a pair of nodes 1500 meters away, even a short beacon message will take more than 2.5 seconds to travel between them, including a 1 second propagation delay and a larger than 1.5 second transmission delay. On the other hand, currents and tides might cause frequent changes in UAN topologies, which can lead to a non-trivial overhead to maintain the neighbor information. For the same reasons, cluster based methods are impractical for UANs since they also heavily depend on the topology information to select rebroadcasting nodes.

**Broadcast in UANs** In UANs, to counter the high error probability of acoustic channels, coding is commonly employed. Network coding has been applied to UAN like networks featuring time division duplex channels [35]. Besides, fountain coding is a popular candidate due to its rate-less nature and low decoding complexity [9]. Hybrid ARQ, which combines FEC coding and Automatic Repeat-reQuest (ARQ) has also been employed to guarantee reliability.

Despite the benefits brought by coding and Hybrid ARQ, there are still some issues. First, the above schemes are generally hop by hop advancement approaches. The data dissemination will not advance to the next hop until all the nodes in the current hop have successfully recovered all the data packets. This mechanism may cause inefficiency in a heterogeneous network, where within a single hop, the link qualities between multiple receivers and the sender may vary dramatically, especially when there exists a bottle neck link. The bottle neck link might cause a lot of retransmissions even though the majority of the nodes in that very hop have already completed data recovery.

Second, the above schemes overlooked the overhearing opportunities in the network. UANs are by nature broadcast networks, where neighbors can overhear packets being transmitted. In a multi-hop UAN, a node can have up to three rounds of overhearing opportunities: the first one comes from the senders in its prior hop; the second one originates from its neighbors in the same hop and the last one stems from the nodes in its next hop. These overhearing opportunities provide a great data recovery chance when combined with network coding [36].

**The Proposed Solution** To address the above issues, we propose TBS, a two-phase broadcast protocol composed of the Fast Spreading phase and the Data Recovery phase. The key goal of the first phase is to spread the data packets throughout the network as efficiently as possible. The nodes take full advantage of overhearing opportunities to accumulate encoded packets. Once sufficient encoded packets are accumulated, the nodes are able to recover all the original data packets with the help of network coding.

The Fast Spreading phase cannot achieve 100% reliability. In case a node fails to recover all the original data packets, it will step into the Data Recovery phase. Request will be sent and correspondingly neighbors will coordinate to send responses. A key idea in this phase is that a node delays sending out request until it estimates that all the nodes within the range of three hops have completed the Fast Spreading phase. With this scheme, the Data Recovery at a node does not interfere with ongoing data traffic in other areas of the network.

**4.2 System Model**

TBS targets a multi-hop broadcast UAN, as shown in Fig. 13. There exists a source node trying to send a block of data packets to every node in the network. Since TBS requires no topology or neighbor information, the targeted UAN can be either a static one or a dynamic one with varying topology due to node movements, currents and tides.

A UAN node is equipped with an underwater acoustic modem to send and receive data packets. Acoustic modems are half duplex and can only be in the sending or receiving status at one time. Acoustic modems are
associated with several types of delays such as processing delay and state transition delay [8]. In the design of TBS, we focus on the following types of delays:

- Transmission delay: Traditionally transmission delay is only related to the length of the packet to be transmitted and can be calculated as:
  \[ D_{tx} = L \cdot 8/R \]  
  Here \( L \) is the packet length and \( R \) is the modem transmission rate. However, [26] observed the long preamble issue associated with acoustic modems stemming from the process of synchronization, signal detection, automatic gain control (AGC) control and channel estimation. The preamble length of Benthos modems is measured to be 1.5 seconds and that of OFDM modems [5] is observed to be 0.66 seconds. Correspondingly, we have to modify the transmission delay calculation to be:
  \[ D_{tx} = L_{pre} + L \cdot 8/R \]  
  where \( L_{pre} \) is the preamble length of acoustic modems.

- Decoding delay: Decoding delay refers to the delay associated with the process of data reception and demodulation. According to [37], it takes 170 ms to receive and demodulate a packet of 80 bytes. Here without confusion, we term the 170 ms as the decoding delay. Therefore, for a block of \( N \) data packets with length equal to \( L \), the decoding delay is:
  \[ D_{de} = N \cdot L \cdot 0.17/80; \]  

- Propagation delay: Propagation delay has been well understood by the research community and is decided by the distance between two network nodes and the sound speed. The sound speed is usually affected by temperature, salinity and some other factors [38]. Here we assume the sound speed to be constant and the propagation delay is drawn as:
  \[ D_{pr} = D/V \]  
  where \( D \) is the distance between two nodes and \( V \) is the sound speed.
4.3 Protocol Design

The Fast Spreading Phase: TBS essentially tries to answer three questions in the Fast Spreading phase: whether to rebroadcast (whether a receiver should be selected as a forwarder), and if yes, when and how many packets to rebroadcast. The first one aims to alleviate the broadcast storm problem by limiting the size of the forwarder set. The latter two target at reducing collisions at a receiver in order to improve efficiency.

Whether to rebroadcast: Since it is assumed that no topology or neighbor information of the network is available, TBS adopts a probability based forwarder selection scheme. The probability that a node rebroadcasts after receiving a group of encoded packets is:

\[ P_R = \frac{K'}{K} \cdot e^{-\frac{P_i - P_r}{T_{de}}} \]  

(6)

where \( K \) is the block size; \( K' \) is the number of linearly independent encoded packets the node has accumulated so far; \( P_i \) is the initial power level of the node; \( P_r \) is the remaining power level of the node.

The retransmission probability is proportional to the number of linearly independent encoded packets accumulated so far. A node can only be in the sending or receiving status at one time due to the half duplex underwater acoustic modems. Intuitively, if a node has already got a decent amount of linearly independent encoded packets, it should serve as a forwarder and contribute to packet accumulation at its neighbors. Although it gives up the overhearing opportunity at this round, it is still possible to recover the complete data set with the next overhearing opportunity from its next hop neighbors. On the contrary, if a node has only accumulated a small number of encoded packets so far, it should give up the chance of serving as a forwarder. In this way, it can stay in the receiving state and accumulate more encoded packets.

The second factor affecting the retransmit probability is the remaining power level of a node. For underwater acoustic modems, sending a packet always consumes much more power than receiving one [37]. Therefore, we take into account the remaining power level of a node to meet the energy constraint challenge of UANs.

When and how many to rebroadcast: After a node is selected as a forwarder, we have to carefully schedule when and how many packets it rebroadcasts. Otherwise, if all the forwarders at the same hop rebroadcast simultaneously, it will cause collisions at the receivers.

The collision at a receiver originates from the long decoding delay of acoustic modems. For instance, in Fig. 13, let us suppose Node A2 and A3 are both rebroadcasting. When the packets from Node A2 reach Node B3, it will take the acoustic modem a period of time to decode, as can be estimated by Equation 4. During this decoding period, packets arriving at Node B3 from Node A3 will be discarded.

To avoid the above situation, we propose a scheduling algorithm in terms of how many packets and when to rebroadcast. Every forwarder retransmits exactly \( K \) encoded packets, which is equal to the block size. If a forwarder has already recovered the \( K \) original data packets, it will retransmit \( K \) linearly independent encoded packets. If a forwarder has only got \( K' \) (\( K' < K \)) independent encoded packets, it will re-encode the \( K' \) packets to generate another \( K - K' \) packets and retransmit in total \( K \) encoded packets as well.

The uniform number of encoded packets to rebroadcast makes it easier to schedule retransmit timings among forwarders without causing collisions. A forwarder has to perform a proactive back-off before rebroadcasting and the back-off time is:

\[ T_{BF} = D_{de} \cdot [P_e \cdot M] \]  

(7)

where \( D_{de} \) is the decoding delay of \( K \) encoded packets as calculated in Equation 4; \( P_e \) is uniformly distributed in \([0, 1]\); \( M \) is the maximum back-off number. \( M \) is a system parameter of TBS that is usually decided by the node density or number of neighboring nodes. How to determine the optimal \( M \) without knowing the above prior information is a future work for TBS.

This scheduling algorithm cannot guarantee zero collision but can reduce the collision chance. As an example, in Fig. 13, we assume that Node A1, A2 and A3 are selected to be forwarders and retransmit to Node B3. In the ideal case, Node A1, A2 and A3 respectively back-off for 0, 1 and 2 \( D_{de} \). The three forwarders all transmit \( K \) packets and therefore yield the same transmission delay \( D_{tx} \). Here we ignore the propagation delay difference between the three forwarders to Node B3 and assume a uniform propagation delay \( D_{pr} \). The reason is that the transmission delay is usually much more dominant than the propagation delay, which has been proved by real world sea experiment [22]. Therefore, after a time period of \( D_{tx} + D_{pr} \), the three blocks of \( K \) encoded packets from these three forwarders will arrive at Node B3 in sequence with \( D_{de} \) as the interval between every two
consecutive blocks and therefore no collision is incurred. This conclusion stands even when packet loss occurs on some links. If we do have to consider the propagation delay difference, we can add a guard time to Equation 7 to ensure the correctness of this scheduling algorithm.

The Data Recovery Phase If a node fails to recover all the $K$ original data packets in the Fast Spreading phase, it will get into the Data Recovery phase. The Data Recovery phase is essentially a Hybrid ARQ procedure. The node broadcasts a request for more encoded packets. Upon receiving the request, a neighbor performs a proactive back-off as what a forwarder does in the Fast Spreading phase and then broadcasts $K$ encoded packets. If a neighbor overhears a broadcast before its back-off timer expires, it will cancel its scheduled broadcast.

One critical goal of this phase is to guarantee that the Data Recovery phase at a node does not interfere with the Fast Spreading phase at other nodes. Fig. 14 illustrates this idea. Suppose Node A at hop n-2 is in the Data Recovery phase and Node A' at hop n-1 is sending response to A (The distance of a hop is equal to the transmission range of acoustic modems); in the meanwhile, Node C at hop n+1 is in the Fast Spreading phase and rebroadcasting encoded packets. In this case, the packets sent from A' and C will probably collide at Node R at hop n.

Therefore, to avoid the interference between the two phases, we let a node delay sending out the request until it estimates that all the nodes within the range of three hops have completed their Fast Spreading phase. After a node receives its first packet, its estimated delay before sending out the request is:

$$T_{BD} = 4 \cdot (D_{tx} + D_{pr} + M \cdot D_{de})$$

However, we notice that due to the probability based rebroadcast, there exists a chance that a node receives no packet in the Fast Spreading phase and therefore does not have a time point to start the delay timer. Inspired by the dual band broadcast scheme proposed in [39], a possible solution is to employ a second high power and long distance signal as the broadcast initialization notification. Upon receiving this signal, a node is aware of the starting of a broadcast and can thus set up its delay timer. Another benefit of this scheme is that with this notification, a node knows when its Fast Spreading phase begins and therefore can estimate when its Fast Spreading phase ends since the Fast Spreading phase contains at most three rounds of overhearing.

4.4 Preliminary Simulation Results

We will investigate the behavior of TBS, especially the efficiency of the combination of opportunistic overhearing and network coding, which can be evaluated by the number of nodes getting into the Data Recovery phase. We will also compare TBS with another two baseline broadcast protocols. We have implemented an FEC coding based Hop-by-hop (FCH) broadcast scheme. FCH employs a Hybrid ARQ scheme to guarantee reliability within one hop. After all the nodes in the current hop have completely recovered all the $K$ original data packets, the broadcast will advance to nodes in the next hop. We have also implemented a pure probability based broadcast scheme. To alleviate the broadcast storm problem, every node does a coin toss to decide whether it will rebroadcast.

We simulate two UANs both with 73 network nodes: one with a uniform topology and one with a random topology, as shown in Fig. 15 and Fig. 16 (The red node is the source node). Unless otherwise specified, the network settings are as follows: the block size is 50, namely the source node has 50 packets to broadcast; each packet is 80 bytes long; the decoding delay for a packet is 170 ms; the average distance between two nodes is 1500 m; the average Packet Error Rate is 0.3.

TBS Behavior In this section, we focus on analyzing the efficiency of combining opportunistic overhearing and network coding, which can be measured by the number of nodes forced into the Data Recovery phase.

First, we investigate the impact of PER. As shown in Fig. 17, the number of nodes out of the 72 receivers getting into the second phase of TBS grows larger with an increasing PER in both the uniform and random topology. When PER is small, only a small number of nodes get into the second phase, which means that most of the nodes can recover all the data packets in the Fast Spreading phase. However, with a larger PER, due to the high packet loss rate, a node usually cannot accumulate enough encoded packets from the Fast Spreading phase. Also we note that the number of nodes getting into the Data Recovery phase is smaller in the uniform
topology than in the random topology. The reason is that nodes in the uniform topology usually have multiple neighbors and the numbers of their neighbors are close to each other. This means that every node can get good overhearing opportunities from its neighbors and enlarges the chance of complete data recovery in the Fast Spreading phase. By contrast, in a random topology, some nodes have a decent amount of neighbors while some node may only have a single neighbor. The latter one can receive only a limited number of or no encoded packets in the Fast Spreading phase and usually ends up being in the Data Recovery phase. However, we have to note that a larger number of nodes into the Data Recovery phase does not necessarily mean a larger network-wide broadcast completion time. The reason is that in TBS, the Data Recovery phase at some nodes can parallel with the Fast Spreading phase at other nodes three hops away or further.

We also investigate the impact of block size. As shown in Fig. 18, block size has no impact on the number of nodes getting into the Data Recovery phase.
Network-wide Broadcast Completion Time In this section, we investigate the network-wide broadcast completion time with the three protocols. Network-wide broadcast completion time is equal to the delay between when the source node broadcasts a block of encoded packets and when all the nodes in the network have completely recovered the data block.

Fig. 19 and 20 show the broadcast completion time of the three protocols in the uniform topology respectively with a low PER (0.3) and a high PER (0.6). TBS achieves the smallest broadcast completion time. On the one hand, the combination of opportunistic overhearing and network coding in the Fast Spreading phase improves the broadcast efficiency. On the other hand, the parallelization of the Fast Spreading and the Data Recovery further decreases the network-wide broadcast completion time. FEC coding based hop-by-hop advancement yields a larger completion time due to the under-utilization of overhearing opportunities. Pure probability based method leads to the largest broadcast completion time due to the inefficiency of the predetermined rebroadcast probability. The simulation results with the random topology in Fig. 21 and 22 follow the same trend and also demonstrate the advantages of TBS. All the three protocols achieve 100% reliability.

4.5 Summary

In this work, in order to address the reliable broadcast problem in UANs, we propose a Two-phase Broadcast Scheme (TBS), which does not depend on topology or neighbor information. The Fast Spreading phase combines opportunistic overhearing and network coding to improve the broadcast efficiency. TBS addresses whether to rebroadcast, how many to rebroadcast and when to rebroadcast in order to alleviate the broadcast storm and reduce collisions. The Data Recovery phase aims at ensuring 100% reliability. By deliberately delaying sending request, the Data Recovery phase at a node will not interfere with the Fast Spreading phase at other nodes.

5 Conclusion and Future Work

In this prospectus, we tackle the reliable and efficient data transfer problem for Underwater Acoustic Networks. First, we propose a CCRDT protocol dedicated for multi-hop string topology UANs. CCRDT combines GF(256) Random Linear Coding and selective repeat to guarantee the per-hop reliability. A multi-hop coordination scheme is employed by CCRDT to enable transmission pipelining and eliminate collisions. Second, we propose SASHA, which is a coordination based MAC protocol for UANs with arbitrary topologies. SASHA embraces most commonly employed techniques in coordination based MAC design. More importantly, a sea test was conducted to study the behavior and performance of SASHA. Important issues have been pointed out and
corresponding design guidelines are given. Last, we design TBS for reliable and efficient broadcast in UANs. TBS does not rely on topology or neighbor information. It includes the Fast Spreading phase focusing on efficiency and the Data Recovery phase focusing on reliability.

For CCRDT, we plan to implement an adaptive scheme to estimate the time slot length, which can dynamically adjust according to the changing PERs. This will further improve the adaptability and efficiency of CCRDT. Also we plan to set up experiments to evaluate the performance of CCRDT in real ocean tests.

For SASHA, we want to add a topology probing component to SASHA to make it aware of the accurate network topology. In this way, the large queuing delay with SASHA partly imposed by transmission range uncertainty is expected to be reduced.

For TBS, first we would like to design a more effective scheduling algorithm in the Fast Spreading phase. Currently, TBS cannot decide the optimal maximum back-off time for a forwarder. Also we hope to more intensively and accurately evaluate the performance of TBS, especially via lab tests and field tests and therefore we can have a better knowledge about how TBS performs in real world.

References

