A Cluster Based Delay Tolerant MAC Protocol for Underwater Wireless Sensor Network

Zhanyang Zhang
Computer Science Department, College of Staten Island/City University of New York
Staten Island, New York, USA

Abstract - the propagation speed of an acoustic signal is much slower than the speed of a terrestrial radio signal due to the physical characteristics of an underwater acoustic channel. This large delay can impact the throughput of the channel. There is also a very high delay variance which presents unique challenges to the designs of efficient protocols. The inconsistent delay renders many traditional communication protocols insufficient since they rely on accurate estimations of the round trip time (RTT) between two communication nodes. We presented a unique approach to develop a Cluster based delay-tolerant protocol (CBDTP) to address these problems by predicating a value for a sensor node if its data were not received at the sink node instead of having the sensor node retransmit its data. The CBDTP can reduce data traffic over the networks and uses the network resource more efficiently. In this paper we improved our CBDTP protocol by introduced a self-adoptive algorithm at MAC layer protocol to better address the inconsistent delay problems. It improves the performance of CBDTP protocol in a dynamic deployment environment.

Keywords: Underwater Wireless Sensor Networks, Acoustic Channel, Delay Tolerant Protocols.

1 Introduction

We live on the surface of the earth of which close to 70% is covered by water. Underwater wireless sensor network (UWSN) is a promising technology to enable applications such as: oceanographic data collection, pollution monitoring, offshore exploration and tactical surveillance systems for homeland security. Ocean bottom sensor nodes can collect, process, and transmit data to surface or onshore stations via direct link or multi-hop relays. Compared to most terrestrial wireless sensor networks, the major difference is that UWSN uses acoustic wave instead of radio wave communication due to the poor propagation capability of radio waves under water. The physical characteristics of the underwater acoustic channel present a new set of challenging problems, such as, low bandwidth, long and variable propagation delays (5 times longer than radio) and high failure rate [1]. Many terrestrial radio-based wireless network protocols might not work well for underwater acoustic sensor networks.

Our research focuses on a major type of underwater wireless sensor network application, underwater environment monitoring, and security surveillance in rivers, lakes and oceans. This type of network requires a long duration of reliable operation with consideration of sensor failures and intermittent connections. Our goal is to explore new concepts in network protocol design that lead to more efficient use of scarce resources while tolerating longer delays and disrupted connections.

The rest of the paper is organized into the following sections. In section 2, we summarize some of the key issues and challenges in protocol design and implementation for underwater sensor networks due to long and inconsistent delay time. We reference some of the recent works related to delay-tolerant protocols to provide a background for our research. In section 3, we define a system model with abstraction and assumptions about the operation environment for the type of UWSN applications we are studying. In section 4, we present the design philosophy of the CBDTP as well as its finite state machine representation and the algorithms used in the CBDTP protocol including a newly designed self adoptive algorithm to adjust the expected packet arrival window based on past delay profiles. Simulation study and performance analysis are presented in section 5. We conclude the paper in section 6 to highlight the major contributions of our study and some open problems for future research.

2 Background and Related Work

Recent advances in communications and electronics have enabled the development of economical, low-power, multifunctional underwater sensor nodes that are small in size and can communicate for short distances via wireless links [2,3]. These sensor nodes are usually scattered underwater at different depths or at the bottom of the ocean floor. Each of these sensor nodes has the capability to collect data and route data back to the surface or to onshore sinks. Data is routed to the sinks by a multi-hop infrastructure-less architecture. Then the sink sends the data to a data processing center that could be many miles away [1]. One of the typical underwater wireless sensor network deployments is shown in Figure1.
Figure 1 shows a UWSN network that has a 3-D topology and consists of heterogeneous nodes. Some nodes are more powerful and can perform more functions than others. For example, there are one or more sensor nodes (uw-sensor) in each cluster. But there is only one cluster head (uw-sink) which has more resources and can perform more functions than the sensor nodes, such as collecting data, controlling sensor nodes, and relaying aggregate data to the surface sinks via a vertical communication link. A sensor node only performs a few simple tasks with limited resources. Most sensors have the components that provide data collection/storage, data processing, communication functions and power supply (Fig. 2).

Figure 2. Internal architecture of an underwater sensor node

These nodes can be deployed from a few feet to 10,000 feet deep underwater. In the case of an oil company’s offshore exploration and production platforms, sensor nodes are deployed up to 12,000 feet deep. These sensors can cover 400 square km of ocean floor [3]. They are used to collect data about water temperature, ocean current motion, strain gauges, long term fatigue and pipe corrosion.

The underwater acoustic propagation speed is five orders of magnitude slower than the speed in a terrestrial radio channel. The speed of acoustic signal is approximately 1500 m/s [2]. This large delay can reduce the throughput of the channel. There is a very high delay variance which makes efficient protocol designs even more difficult. The inconsistent delay renders many traditional communication protocols insufficient since they rely on accurate estimation of the round trip time (RTT) between two communication nodes.

Let’s consider TCP protocol as an example. TCP is a reliable communication protocol between nodes. Node A sends a packet to node B. Node B sends an acknowledgement (ACK) back to node A when it receives the packet. If node A did not receive the ACK from node B after waiting for a period of time (based on RTT estimation), node A will assume the package was lost and it will send the packet again. This hand-shaking protocol guarantees a reliable communication between the two nodes. But the traditional TCP protocol does not work well in the case of long and inconsistent delays for underwater acoustic networks due to high delay variance and disrupted communication link. Considering the same example above, node A does not know if the packet were lost or how long it should wait for the ACK from node B. There is no way to know how long the RTT will be in the case of an extremely long delay. Therefore node A will keep retransmitting the same packet many times. It overloads the...
channel, wastes resources, and increases the risk of packet collisions.

There are a few previous published papers on delay-tolerant networks addressing the long delay problems in underwater acoustic networks. In [4], a propagation-delay-tolerant collision avoidance protocol was introduced to improve throughput performance while avoiding collision. The authors of [4] conducted a simulation study to show their protocol has better performance than traditional CSMA/CA protocol which is a widely used in Wi-Fi networks. The authors of [5] address the long delay problem in acoustic networks from network routing perspective. In [6], a data transport ferries concept was introduced as a delay-tolerant solution to address both long delay and low channel reliability problems. A data ferry is used to store and forward data from one node to another node. It works better in case of long delay and disrupted connection between certain nodes. Their work achieved a common goal, which is overcoming or tolerating the long propagation delay. Their proposed protocols still rely on a fair accurate round-trip-time (RTT) estimation. They do not work well in case of high delay variance since it would be difficult, if not impossible, to fair accurately forecast the RTT value in such case.

We have developed a prediction based delay-tolerant protocol (PBDTP) to address this problem at TCP layer [7]. PBDTP can tolerate long and inconsistent delays as well as connection disruptions by predicating a value for a sensor node if its data was not received at the sink instead of asking the node to retransmit its data. PBDTP significantly reduces data traffic over the networks and uses the network resource much more efficiently. In this paper, we present our recent study on applying PBDTP concept in designing a cluster based delay tolerant protocol (CBDTP). CBDTP is an enhancement to the most commonly used CSMA/CA MAC layer protocol (IEEE 802.11). Our simulation study shows that it increases the throughput and reliability of communications within a cluster.

3 The Cluster Based Network Model

The system model we specified here reflects the operation of a typical UWSN application for environment monitoring and detection. The system model is constructed based on acoustic channel characteristics that are applicable to underwater monitoring applications from a few hundred feet to ten-thousand feet. These UWSN applications are deployed in oceans or lakes with a large body of water. It is a heterogeneous sensor network with a three dimensional hieratical topology as depicted in Figure 1. A basic underwater operation unit is a cluster which consists of a number of sensor nodes with less resources and one cluster head node with more resources (Fig. 3).

Sensor nodes associate themselves to a cluster according to their close proximity to the cluster head. In most of the case, sensors in a cluster are located at relatively the same depth. Sensors only communicate with their head node directly (in one-hop distance). The cluster head collects data from sensor nodes or issues control commands to sensor nodes via a half-duplex acoustic channel. The cluster head also sends data to the surface sink vertically via one hop or multiple relay nodes. Due to effective communication ranges of the nodes, there may be a number of clusters in order to cover a large underwater area. In deep water applications, it may be necessary to deploy multiple relay nodes at different depths to relay data to a surface sink in multiple hops.

To keep the system model simple, we assume there is a direct link between cluster head and surface sink (Fig. 3). In most of the case, the cluster head has more resources and transmission power to send acoustic signal over much longer distance than a sensor node. We further assume that all nodes are stationary (anchored without mobility, except for drafts in a short range due to undersurface water current). Sensor nodes are deployed by surface or submarine vesicles. The vertical position (depth) is determined by the length of the anchor chains. We assume the sensors' horizontal positions are known to the UWSN operators via localization techniques [8]. However, the underwater horizontal positions may vary from their targeted positions due to drafts as a node is sinking. We assume such drafts are in small range and can be negligible.

From a real-world UWSN operation perspective, we describe how the system model works in 3 phases as listed below:

- **Pre-deployment configuration** - Before sensor nodes are deployed underwater, sensor node IDs are registered. Each sensor node needs to know the cluster head ID (CID) and the cluster head needs to know the entire sensor node IDs (SID) which belong to the cluster.
- **Cluster formation (convergence)** - During this phase, each sensor will exchange a probing message with the cluster head in order to confirm the existence of the
sensor node and to profile the communication link at the cluster head.

- Normal Operation – The cluster head starts to collect data from the sensor nodes and occasionally exchanges control/status messages with sensor nodes after forming the cluster. The system model can support three major groups of communication protocols for data communication between sensor nodes and the cluster head node, namely, a TDMA based protocol, a cluster head based query protocol (poll), and a sensor node event-driven protocol (push).

In this paper, we present our initial work on the cluster head based query protocol. We investigated this protocol first because of its simplicity and its wide applications in UWSN environment monitoring. We will report our work on TDMA and event-driven protocols in the near future.

4 The Protocol Design

Underwater communication channels are subject to acoustic propagation characteristics including higher rate of link failure, long and inconsistent propagation delays [9]. Our design objective is to develop protocols that can tolerate these challenges presented in UWSN to improve performance and conserve system resources.

Since sensor nodes and the cluster head node in a cluster exhibit different behavior at different phases, it is better to illustrate the CBDTP protocol using two finite state diagrams, one for sensor nodes and another one for cluster head nodes [Fig. 4].

There are 6 states for the cluster head nodes (Fig. 4b), they are:

1. **Idle state** - This is the initial state for cluster head node. It works in the same way as in the case of sensor nodes.
2. **WaitConvReg state** – This is the state where a cluster head node listens for the probe messages coming from all the sensor nodes. It sets a timeout period that controls how long it will wait for the probe messages.
3. **NoteReg state** – When the cluster head node gets a probe message, it enters the NoteReg state to register the sensor node ID in its registry. Then it checks the timeout to determine if it should return to the WaitConvReg state for the next probe message or return to the Idle state and set the convBit=1.
4. **SendingSRP state** – While in the Idle state, if convBit=1, the cluster head node enters this state. It will send a data request packet (SPRSent) to the sensor node which is in front of a registered sensor node queue. Then the cluster head node enters Wait state and sets up a timeout clock.
5. **Wait state** – While in the Wait state, a cluster head node waits to receive the data packet from the sensor node. It will log the data (DataRecved) if the data packet arrives within the timeout period. Then it returns the SendingSRP state and repeats the same cycle with the next sensor node in the queue. Otherwise it enters the Predict state.
6. **Predict state** – A cluster head node enters this state because it did not receive the expected data packet from the target sensor node within the timeout period. The data packet could be lost or it has taken an unusual long delay. The cluster head node predicts the data value based on previous data from this sensor node. Then it returns the SendingSRP node.

There are 6 states for the cluster head nodes (Fig. 4b), they are:

1. **Idle state** - This is the initial state. All the sensor nodes are set to this state at the time of deployment. The converge bit is set to zero (convBit=0)

2. **SendingCRP state** – If a sensor is in the Idle state and convBit=0 then it enters the SendingCRP state. This is the state where sensor nodes send a probe message (CRPSent-cluster register packet) to the cluster head.

3. **WaitingConverge state** – After sending a probe message, a sensor node enters the WaitingConverge state to wait the reply from the cluster head. Upon receiving the reply (REC_ACK), the sensor node returns to the Idle state and set the convBit=1.

4. **ReceivingSRP state** – While at the Idle state, if convBit=1 then a sensor node enters the ReceivingSRP state. In this state, a sensor node waits for the data request packet from the cluster head node. When the sensor node receives the data request (ReqRecved), it enters the DataSending state.

5. **DataSending state** – While in this state, a sensor node assembles a data packet then it sends the data packet to the cluster head node. Afterward, it returns to the ReceivingSRP state and repeats the same cycle.

![State diagram for sensor nodes](image1)

(a) State diagram for sensor nodes

![State diagram for cluster head nodes](image2)

(b) State diagram for cluster head nodes

Figure 4. Finite state machine representation of CBDTP protocol

There are 5 states for the sensor nodes (Fig. 4a), they are:

1. **Idle state** – This is the initial state. All the sensor nodes are set to this state at the time of deployment. The converge bit is set to zero (convBit=0)
The proposed CBDTP protocol design has two key components. Here we describe briefly how each component works.

4.1 The Predication Algorithm

As shown in Figure 1, sensor nodes are grouped in clusters. Each cluster has a cluster head (black dot) and a few sensor nodes. Sensor nodes send their data to the head node via one-hop horizontal links. The head node sends the aggregated data to a surface sink via a vertical link (Fig. 3). The head node collects data from sensor nodes in a round-robin style at fixed cycles (called rounds). The data flow is from sensor nodes to the cluster head then to the surface sink node. The control flow is the reverse of the data flow. Since both data and control share the same half-duplex link, the link is shown as bi-directional in Fig. 3.

Published research results show that the delay variance in horizontal acoustic links is generally larger than in vertical links due to the nature of underwater links [10]. The links between sensor nodes and the cluster head node are weaker links due to this fact. In the case of a long delay or an unexpected interruption, the head node will predict a value for the sensor nodes using the prediction algorithm. Based on our observations, we make two assumptions, (1) a sensor node most likely will report the same value or a similar value with a small change over a short time window. (2) Adjacent sensor nodes will most likely report the same values or similar values with small changes. The prediction algorithm predicts a value for a sensor node either based on the previous values of the sensor node, or its neighbor sensor nodes’ values. It also can predict the value based on a combination of both. The lost data packets are compensated at the cluster head node with the predicted values without resending the data. The CBDTP protocol can tolerate long delays and irregular disruptions with efficient usage of network resources.

4.2 The Adjustment Algorithm

If the head node receives data from its sensor nodes after making the prediction, it can replace the predicted value and use the real data value as the basis for future predictions for better accuracy. If the difference between the received value and predicted value exceed certain thresholds, the head node will send the newly received value to a surface sink so that the sink can make necessary adjustments based on application requirements. Otherwise no actions are taken.

5 Simulation and Performance Analysis

A preliminary simulation study was conducted to validate the system model and to measure the CBDTP protocol performance against our design objectives. We measure the communication link performance between a sensor node and the cluster head in term of throughput (number of packets per unit of time) and the data accuracy (the difference between the data sent by sensors and estimated data received at the sink in the case of predictions were made).

We compared the performance of the CBDTP protocol and the traditional CSMA/CA protocol under the same simulation conditions to test if the proposed CBDTP protocol can achieve reasonable performance gains over CSMA/CA protocol. We implemented a simulation study at MAC layer to compare the CBDTP protocol with a few other protocols including CSMA/CA protocols [3, 4, and 5].

5.1 Simulation Model

There are many ways to conduct a simulation study. In this paper, we present a set of simulation scenarios that closely reflect the class of UWSN applications for environment monitoring and detection. Since a cluster is the basic operation unit of a UWSN application and it presents the most challenges in its horizontal communication links, therefore we focus our simulation study on communications links between sensor nodes and the cluster head node within a cluster. We set the cluster size with 7 sensor nodes and one cluster head node. All the sensors are within the maximum distance of 500 meters from the cluster head. The baseline propagation delay (dt) is about 1/3 second or 333ms. The propagation delay variance (Δdt) can be 0 to 100ms in addition to the baseline. In order to simulate the reliability factor of the channel, we introduce a probability of channel failure (p). We set the channel failure value in the range of (0.01, 0.05, 0.10, 0.15, and 0.20) to study the impacts on channel performance.

The simulation program is written in C++ with distributed processes to emulate the sensor nodes and the cluster head node. Each process is a self-control entity. The sample size of each simulation run is 1500 rounds of data collection.

There are two specific functions that we implemented in the simulation.

- We adopted the Newton method to estimate the data value in the prediction algorithm using three data values in previous rounds to achieve better accuracy.
- Due to propagation delay inconsistency, a fixed timeout window set for an acoustic channel may not best reflect the dynamic changes over time. We introduced a dynamic self-adoptive timeout window management schema based on the well known Jacobson-Karn’s Algorithm [11] which adjusts the timeout window based on past delay profiles.
5.2 The Adaptive Round Trip Time (RTT) Estimation Algorithm

The Jacobson-Karn’s Algorithm in its original form is defined as:

\[ D = \alpha D + \beta |RTT - M| \] (1)
\[ \text{Timeout} = RTT + 4D \] (2)

Where RTT is the expected value for round trip time. D is the standard deviation of RTTs. M is the observed value for round trip time. \( |RTT - M| \) is the difference between the expected and observed values. In our simulation we set \( \alpha = 1 \), \( \beta = 1/8 \) and use \( 2D \) instead of \( 4D \) for a better RTT estimation pertains to unreliable underwater channels.

There is a known problem with Jacobson-Karn’s Algorithm when it is applied to a highly unreliable channel. When expected data does not arrive during the timeout window, it is not clear whether it is due to congestion or due to lost data. Making a wrong judgment can be costly. For example, increasing timeout window may improve performance in case of congestion. But it can impact performance negatively in case of data lost. To address this problem in our simulation, we developed an algorithm to offset the timeout window calculated by Jacobson-Karn’s algorithm with a dynamic factor based on past history of data lost ratio.

5.3 Performance Analysis

Our initial simulation results show that the proposed CBDTP protocol performs better than traditional CSMA/CA protocol in the above scenarios that are used for the simulation. Figure 5 shows the throughput (data packet received by the cluster head node per minutes, 500 bits/packet) in a typical environment monitoring application with low to moderate data rate. There are three lines in the chart. They represent the effective throughput (square-dot line), the real throughput (diamond-dot line) and the CSMA/CA throughput (triangle-dot line). We use both the effective throughput and real throughput to measure the performance of the proposed CBDTP. The effective throughput includes the predicted data packets which were lost during transmission. Since the CBDTP uses the estimated data value to make up the lost data packet. The net effect of throughput is higher than the real throughput which only counts the packet that went through the channel. The performance gain over the CSMA/CA protocol is amplified as the channel quality gets worse (as channel failure probability increases from 0.01 to 0.20). In other word, the CBDTP protocol can better tolerate the channel quality degradation than the CSMA/CA protocol does.

While the CBDTP protocol has a notable performance gain in terms of throughput, we are also very interested in the quality of CBDTP in terms of data accuracy. We define data error as the differences between the received data values and the original values the sensor nodes collected and sent. In order to measure the data error, we extracted the average value of sensor data from all the nodes, which is the ocean water temperature and the average value of data received at the cluster head node (includes both real data and estimated data). Our simulation data shows that the two average values are very close (Figure 6). The average value of sensor data is shown as square-dot line and the average value of received data is shown as diamond-dot line. To measure data error variance and distribution, our study shows the mean value of data error is 0.12 and the standard deviation is 0.90 with 15000 samples.
6 Conclusions

In this paper, we point out the challenges and problems that UWSN design engineers and operators are facing, in particular, the problems of link reliability, long delay and inconsistent delay. Traditional protocols do not work well with these problems. We proposed a new CBDTP protocol to overcome these challenging problems. We made two key contributions in the proposed CBDTP protocol, namely, data prediction and adjustment algorithms. Our simulation study shows the CBDTP protocol outperforms the traditional CSMA/CV protocol while maintaining good data quality.

There are a few open issues that require future research within the CBDTP framework on TDMA and event-driven protocols as well as further refinement of the query based CBDTP protocol to better manage the dynamic of timeout windows. CBDTP also faces a scalability issue. As the number of sensor nodes increases, the duration of a data collection round increases proportionally. In our simulation, we used a cluster with 10 sensor nodes. But in a real UWSN application, there can be a few hundreds sensor nodes in a cluster. We also plan to investigate CDMA/OFDA multi carrier channels for each sensor node sending data to cluster head at same time in a single RTT window.

7 References


Acknowledgment

This research was supported, in part, under National Science Foundation Grants CNS-0958379 and CNS-0855217 and the City University of New York High Performance Computing Center at the College of Staten Island.