MAC Protocol Implementation on Atmel AVR for Underwater Communication

- Final Report-

Shaolin Peng

speng2@ncsu.edu

Introduction

Underwater acoustic communication is widely used in many areas to collect the data from different kinds of sensors deployed underwater or send control information to remote nodes. However, there are many challenges for underwater communication systems because of the characteristics of acoustic propagation in the underwater environment, such as Doppler spread, ambient noise, fading, high propagation delay, limited bandwidth, wave effect and multipath. MAC protocol, as a developed protocol for Ethernet networks, has been introduced into underwater communication by many researchers and lots of work has been done on the improvement of MAC protocol for throughput and energy efficiency purposes. In general, MAC protocols can be roughly divided into two categories: contention-free protocols and contention-based protocols. Contention-free protocols include TDMA,FDMA and CDMA, where communication channels are separated in time, frequency or code domains. It is common wisdom that FDMA is unsuitable for underwater sensor networks because of the narrow available bandwidth. There are some researches on TDMA and CDMA for underwater networks. However, some problems inherent in these methods have not been well addressed in acoustic networks. For example, the synchronization problem in TDMA and near-far problem in CDMA. Thus, the feasibility of these protocols in underwater sensor networks is unclear. Contention-based protocols includes random access methods and collision avoidance methods. In a random access protocol, e.g., *Aloha,* the sender sends packets without coordination. Thus packet avoidance is totally probabilistic. While in a collision avoidance protocol, the sender and receiver capture the medium through control packet exchange before data transmission. There are many collision avoidance protocols, among which RTS/CTS-based protocols are widely used. The performance of random access methods and RTS/CTS-based approaches in underwater sensor networks is determined by many factors. In this implementation, I developed *Aloha* and RTS/CTS-based *MACA* protocols on a Atmel AVR device in our targeted underwater network and evaluated their performance in the shallow water.

Hardware Overview

The device used for this implementation is a mini transceiver based on Atmega168. This mini transceiver is 1.5 x 2.5 inches, as shown below (Fig 1 &2). It is battery-powered and could transmit and receive up to four different frequencies with a transducer connected. These transducers communicate underwater by ultra-sound. It has three switches, a small display and three LEDs. Atmega168 employed in this implementation is a high-performance, low power AVR 8-bit microcontroller, with 1K RAM and 8K ROM. It has 24 MIPS throughput at 24 MHz.

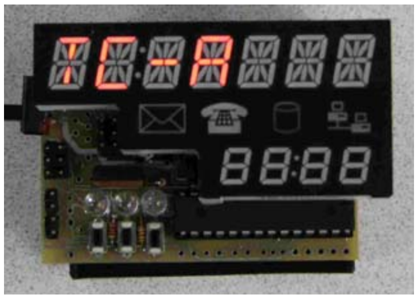


Figure 1. transceiver with display

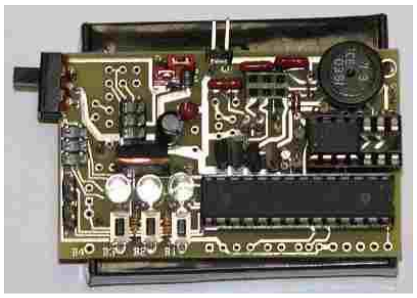


Figure 2. transceiver without display

Protocol Study

There are lots of research on the underwater communication protocols. In [9] the relationship between transmission loss and typical channel parameters, such as sound-speed gradient, sea bottom fluctuation, source depth and propagation distance is studies, which provides a guide to design a practical device design and to choose a suitable protocol. An analysis of the challenges of modeling contention-based medium access control protocols was reported in [4] and it identified several issues complicating such contention-based protocols. This implies that the contention-based protocols with carrier-sense such as CSMA, may not be a good choice considering the nature of the wireless medium. It presented a model to analyze the suitability of Aloha variant protocols. Pure Aloha and slotted Aloha are compared and studied in [5]. Their results were from Qualnets simulator. They indicated that the performance of Aloha and Slotted Aloha is about the same due to that long propagation delay of acoustic signals prohibits the coordination among nodes. The random access and RTS/CTS techniques were studied in [6]. Its results showed that random access is better than RTS/CTS for sparse network with low data rate and non-bursty traffic.

Besides, different from deep water acoustic communication, there are more problems for shallow water communication, such as long delay-spread due to sparse multi-path arrivals and rapid time-varying channel. A much closer study focusing on shallow water was conducted in [8]. It implemented three variant of MAC protocols in data link layer, which are simple Aloha, Aloha with ACK and retries, and MACA using RTS/CTS. From its results, Aloha with ACK has a better performance for smaller packet size (<500 bytes) and less nodes while MACA overcomes for larger packet size and denser networks.

Based on these study on existing protocols, Aloha is more suitable for our case. In our case, the underwater network is relatively small and sparse. The application packet size is not large. As is usually the case in many commercial acoustic modems/transceivers, the physical layer in our device is a half duplex system, which means listening while transmitting is not possible. It does not transmit while reception is in progress and does not receive while transmitting. From the above analysis, Aloha with ACK is a good choice. However, I implemented RTS/CTS-based MACA as well to make a comparison with Aloha.

Protocol Implementation

The basic idea of an Aloha system is simple: senders transmit whenever they have data to be sent. Sender waits for a maximum Round Trip Time (RTT) for an ACK. If there is no ACK back, the sender just waits a random amount of time and resends it. The waiting time must be random or the same frames will collide over and over. At the receiver end, it sends an ACK to the sender if the received packet is error free. Figure 2 is a state diagram. The system will model three layers of the OSI stack - Network, data link, and Physical. Network layer generates the packet to data link layer or gets the packet from data link layer. Data link layer follows the state diagram to process the packets. Physical layer is responsible to encode and send signals, or receive and decode signals.

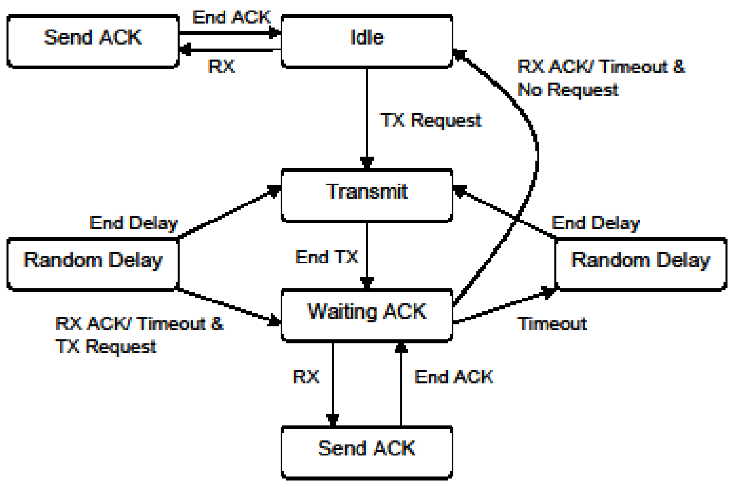
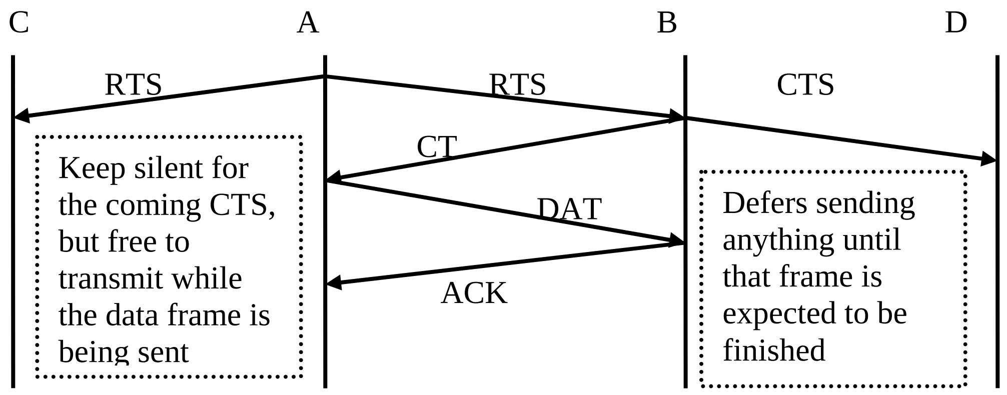


Figure 2. Aloha State Diagram

For MACA protocol the transmitter sends RTS first to the destination, and upon receiving a CTS back, it then sends its data packet to the receiver. And, it requests an ACK after the correct data, which can reduce the waiting time of retransmission. The basic idea is listed below.


C
A
B
D
RTS
RTS
CTS
CT
DAT
ACK
Keep silent for the coming CTS, but free to transmit while the data frame is being sent
Defers sending anything until that frame is expected to be finished

ALOHA protocol pseudo code:

```
if databuffer is not empty
    compose and send a packet;
listening in the channel for a  period (period = wait)
    if received a packet in a period(time<wait)
        if the packet’s destination matches with the its device ID,
            if it is an ACK
                Succeed. Check the databuffer
                if not empty, compose and send next packet
                set ‘wait’ to round trip time
            else if it is a DATA
                if CRC is right
                    put data to RX databuffer
                    compose and send an ACK packet to this node
                else CRC is wrong
                    drop the packet, set ‘wait’ to round trip time
            else if there is an error
                drop the packet, set ‘wait’ to round trip time
        else if the packet’s destination doesn’t match with its device ID
            if it is an ACK
                do nothing
            else if it is a DATA
                keep silent for the ACK to be sent back
    else if period ends (time==wait)
        send the same DATA packet to the same node
        set ‘wait’  to a random time+RTT
```

MACA (RTS/CTS) protocol pseudo code:

```
if databuffer is not empty
    compose and send a packet;
listening in a period (period = wait)
    if received a packet in a period(time<wait)
        if the packet’s destination matches with the its device ID,
            if it is an ACK
                Succeed. Check the databuffer
                if not empty, compose and send next packet
                set ‘wait’ to round trip time
                set status=STATUS_WAIT_CTS;
            else if it is a DATA
                if CRC is right
                    put the data to RX databuffer
                    compose and send an ACK packet to this node
                else CRC is wrong
                    drop the packet, set ‘wait’ to RTT
            else if it is a RTS
                compose and send a CTS to this node
            else if it is a CTS
                compose and send a DATA packet to this node
                set status=STATUS_WAIT_ACK;
            else
                set ‘wait’ to round trip time
        else if the packet’s destination doesn’t match with its device ID
            if it is an ACK
                do nothing
            else if it is a DATA
                keep silent for the ACK to be sent back
                set ‘wait’ to one way time
            else if it is a RTS
                Keep silent for the coming CTS, but free to transmit while the
                data frame is being sent, set ‘wait’ to one way time
            else if it is a CTS
                Keep silent for the coming DATA. Defers sending anything until
                that DATA is  finished by some other node.
                set ‘wait’ to (one way time + data transmission time)

    else if period ends (at time==wait)
        if status == STATUS_WAIT_CTS
            send the same RTS  packet to the same node
            set ‘wait’  to a random time+RTT
        else if status == STATUS_WAIT_ACK
            send the same DATA packet to the same node
            set ‘wait’  to a random time+RTT
```

Packet Format

ALOHA:

Data Packet

ID (4 bits)	Dest (4 bits)	Source (4 bits)	Length (4 bits)	Data (8*n bits)	CRC (8 bits)

ACK

ID (4 bits)	Dest (4 bits)	Source (4 bits)

MACA(RTS/CTS):

Data Packet

ID (4 bits)	Dest (4 bits)	Source (4 bits)	Data (8*n bits)	CRC (8 bits)

RTS/CTS

ID (4 bits)	Dest (4 bits)	Source (4 bits)	Length (4 bits)

ACK

ID (4 bits)	Dest (4 bits)	Source (4 bits)

For the aloha, the data packet has a ID field, which tells the packet type (Data or ACK), a Dest field, which tells the packet destination, a source field, which tells where the packet is from, a length field, which tells the data length, and a CRC field, which is used to check if the data packet is correct. The ACK packet is pretty simple, which only contains ID, Dest and Source.

For MACA protocol, the data and ACK packets are similar as Aloha. The difference is the RTS and CTS packet have a length filed, which is used to notify the destination the data size and notify the other nodes how long they should keep silent.

The CRC in the packet is a Polynomial Code to make sure there is no data error in the packet, otherwise, the sender will send that packet again. Here I use generator $G(x)=x^8+x^2+x+1$ ->(100000111)

I didn't use error correction mechanism in the data link layer, only focusing on the estimation of protocol performance at this initial implementation. However, the convolutional algorithm could be used in the data link layer as the future work.

System Implementation

The operating system or even some schedulers are too complex and expensive for this device. The code size has to be compact due to the limitation of this microcontroller. I use a Round-Robin scheduling for the protocol implementation, with interrupt enabled. The application tasks could be serviced in the loop or be configured as an ISR. They can put the data into TX data buffer. The transmitting task will check if there is any data to send in the data buffer.

The listening task will keep checking the channel if there is a coming packet. If there is a coming packet, the listen task will end earlier, otherwise it will return at the end of the specified period.

The transceiver device is a half- duplex system as is usually the case in many acoustic nodes, which means listening while transmitting is not possible. So when it is transmitting, it shouldn't be interrupted by any events so the interrupt is disabled during that time and incoming packets will be lost due to this half-duplex mode.

Example:

```
While(true){
        if(Databuffer!=Empty)
                Transmit();
        if(TaskReadytoRun)
                Task();
        if(RequestforISR)
                ISRtask();
        ListenChannel();
…
}

Interrupt ISR();
…
```

Transmit and listen are protocol layer functions which will call TX & RX separately. The TX and RX are the interfaces and they will call physical layer functions to send or receive the signals. The Goertzel algorithm is employed in the implementation, to indentify the frequency component from a signal since the transceiver doesn't have any tone decoder. It is implemented in receiver function of physical layer.

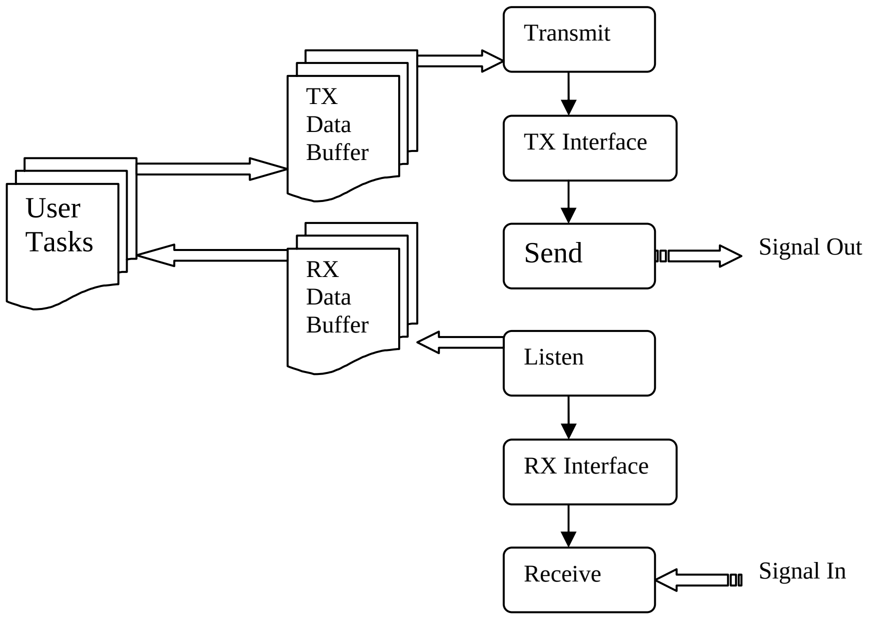


Figure 3. Function Call Graph

Menu System

There is a menu system in the implementation, from which I can easily choose Aloha or MACA to test the performance. It also includes some other functions, such as save data, display results or change configuration. This system is extendable.

EEPROM

In order to save space and reduce the workload on debugging, I put lots of information to EEPROM, such as node ID, Goertzel coefficient table. At the beginning of the program, these values will be read into memory. With the node ID hard coded in the EEPROM, all the nodes could share the same .hex file, otherwise every time the code is modified, I have to create different versions for different nodes. Besides, when the program is done after the tests, I save all the results into a particular address of EEPROM. We can retrieve the results even from a careless power-off or do post-processing later.

Experiments

To estimate the protocol performance in this small network, I focus on the throughput, which is defined as successful received packets divided by total transmitted packets in our case. I built a simple grid network and each node unicasted the packets to all the other three nodes sequentially. In the end, I collected the successfully received packets from all four nodes within a period.

The tests were conducted both indoors and outdoors. For indoor test, I put the transducers next to each other in the same mug. The outdoor test was done in the Lake Raleigh. The transducers were separated about 50~100 feet from each other. In order to estimate the throughput of the protocols, in my test I use the worst case scenario, in which all nodes have their data ready to send from the very begging, and the data is unlimited during the test time. The data speed is 250bps (40ms to send a bit).

Results

Below are the test results.

Lab test:

Aloha Throughput=27.2%

RX packet	TX packet	TXACK	RXACK
34	125	17	14

MACA Throughput=20%

RX Data	TX Data	RX ACK	TX ACK	TX RTS	TX CTS	RX RTS	RX CTS
17	95	13	17	51	36	36	27

Lake test:

Aloha Throughput=8.1%

RX packet	TX packet	TXACK	RXACK
10	123	10	6

MACA Throughput=8.2%

RX Data	TX Data	RX ACK	TX ACK	TX RTS	TX CTS	RX RTS	RX CTS
5	61	3	5	33	9	9	6

From the above results, we can see the overall throughputs for both protocols are not very good because in this worse case scenario, there are lots of collisions when all the nodes have infinite data to send. Both protocols have a better performance for indoors tests than outdoors, the reason is that the situation for the outdoors test is much worse than indoors due to lots of noise and interference introduced by wind, wave and motors.

Clearly, Aloha has a better performance than MACA in lab test but the performance decreases more than MACA when moving to a worse environment, which tells us the interference effects Aloha more than MACA.

Another fact here in these tests is that, during the fixed testing time Aloha could transmit as more as twice data than MACA. So from this point Aloha is a better choice for a small and sparse network.

Some thoughts to improve the protocol

Aloha could be improved to bring more throughput in some cases. For example, if one node gets a packet correctly, we could let it keep waiting for another RTT period instead of sending out its ready data, given the assumption that each time there are more than one packet need to be sent to another node. In this way, there will be less collisions. This will cause temporary starvation and delay the task for one node, but both of the two nodes could finish their transmission tasks in a shorter time.

Current implemented MACA is a little aggressive, because if a receiver gets a data packet but with errors (CRC fails), it will drop the data packet and send its packets if waiting time ends. But we can change the protocol to make the receiver wait for another RTT period in the case that CRC checking fails, because we assume the sender will send the data packet again without an ACK from the receiver. In the way, there will be less collision happening for the next transmission. I did a simple test in the lab by changing the protocol a little bit, and the result came out as follows:

Improved MACA:

RX Data	TX Data	RX ACK	TX ACK	TX RTS	TX CTS	RX RTS	RX CTS
20	65	15	19	43	32	32	26

MACA Throughput=29.0%

By comparing these numbers with the previous MACA, we can see that there is a big improvement on the throughput, even better than Aloha, although the transmitted data amount during the testing period is still less than Aloha.

Another similar idea on MACA is to use NAK as another type of acknowledgement to tell the sender to send the data again. This could help the sender reduce the waiting time in some cases. So basically, by adding new designs, I think MACA could be improved as well to be used in the small and sparse network.

Another observation is, lots of packets are dropped due to CRC error, so if we can correct those errors, then not only we can reduce the collisions, but also we can save more energy. So, in the future work, it is better to implement error correction algorithm for communication, e.g. convolutional encoding (viterbi coding). For the unreliable underwater communication, this could be a great help.

References:

[1] A_an Syed, Wei Ye, Bhaskar Krishnamachari, John Heidemann. Understanding Spatio-Temporal Uncertainty in Medium Access with ALOHA protocols. WUWNeT '07, Sep 2007.
[2] B. Peleato and M. Stojanovic. A MAC Protocol for Ad-Hoc Underwater Acoustic Sensor Networks. WUWNeT '06, Sep 2006.
[3] D. Makhija, P. Kumaraswamy and R. Roy. Challenges and Design of Mac Protocol for Underwater Acoustic Sensor Networks. WIOPT '06, April 2006.
[4] J.H. Gibson, G.G. Xie, Y. Xiao, and H. Chen. Exploring Random Access and Handshaking Techniques in Large-Scale Underwater Wireless Acoustic Sensor Networks. in Proc. MTS/IEEE Oceans Conference 2007, June 2007.
[5] Uichin Lee Luiz Filipe M. Vieira, Jiejun Kong and Mario Gerla. Analysis of aloha protocols for underwater acoustic sensor networks. MobiCom '06, Sep 2006.
[6] Peng Xie, and Jun-Hong Cui. Exploring Random Access and Handshaking Techniques in Large-Scale
Underwater Wireless Acoustic Sensor Networks. OCEANS '06, Sep 2006.
[7] Peng Xie, and Jun-Hong Cui. R-MAC: An Energy-E_cient MAC Protocol for Underwater Sensor
Networks. WASA'07, Aug 2007.
[8] Shiraz Shahabudeen, Mandar Anil Chitre. Design of networking protocols for shallow water peer-to-peer
acoustic networks. OCEANS '05, June 2005.
[9] Xiaomei Xu, Feng Tong, Liuhuai Qing, Yi Tao. Characterization of Wireless Shallow-water Communication Channel Based on Gaussian Beam Tracing. WiCOM '06, Sep 2006.
[10] Zhong Zhou, Zheng Peng and Jun-Hong Cui. Multi-channel MAC Protocols for Underwater Acoustic Sensor Networks. WuWNet '08, Sep 2008.