

An Adaptive Logical Link Layer Protocol for Underwater Acoustic Communication Channels

Daladier Jabba M. and Miguel A. Labrador
Department of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620
Email: {daladier,labrador}@cse.usf.edu

Abstract—Underwater Acoustic Communication (UAC) is characterized by long propagation delays and very noisy channels. Further, both the signal propagation speed and the channel bit error rate are not constant but rather change dynamically depending on the depth and salinity of the water, the activity in the water's surface, the topography of the bottom, and many others factors. This paper builds upon our previous work on logical link control protocols for underwater communications improving the performance of the SW-MER protocol by making it adaptive to channel conditions. In the proposed adaptive version, the protocol exchanges channel quality information to change the number of retransmitted copies, improving the throughput without compromising the reliability of SW-MER. Our simulation experiments with Bernoulli errors and synthetic traces demonstrate that the throughput can be improved in as much as 15% compared with the non-adaptive original version.

I. INTRODUCTION

Undersea exploration in military tactical and civilian applications using sensor networks and swarms of underwater autonomous vehicles (UAVs) is on the rise. Mine-like object detection, shore surveillance, intrusion detection, water quality monitoring, and navigation assistance are common applications. One common aspect of these applications is the need of constant and reliable communication among the sensors and/or UAVs. However, the underwater communication channel is not precisely the best channel. For example, common radio frequency (RF) signals utilized in wireless communications cannot be used underwater, as electromagnetic waves attenuate very rapidly with distance, restricting the communication to only a few meters. In order to overcome this limitation, acoustic communications has been the method of choice for underwater communications.

However, acoustic communications has its own limitations as well. The quality of the Underwater Acoustic Communication (UAC) channel is affected by many different factors when acoustic waves propagate through water. In an UAC channel, the speed of sound is five orders of magnitude lower than the speed of light, meaning that underwater acoustic communications experience very long propagation delays, even in short distances. Further, the speed of sound is not constant, varying with depth, salinity, and other factors. In addition, the underwater communication channel is very noisy, and the

bit error rate (BER) also changes depending on the depth, the activity on the surface, the topography of the bottom, and other factors.

In addition to channel quality issues, another aspect to consider in the design of logical link layer protocols is the fact that transducers for acoustic communication are half duplex. This fact eliminates the possibility of using well-known collision detection mechanisms in MAC layer protocols and the opportunity of employing more efficient sliding window-based ARQ protocols at the logical link control sublayer. As a consequence, the simple and inefficient Stop and Wait protocol is the most widely used ARQ method in underwater networks.

In order to improve the throughput and the reliability of the communication, in [1] we proposed the SW-MER protocol, which included a combination of Stop and Wait and window-based flow control strategy to improve the throughput of the protocol, and an exponentially retransmission strategy to improve the packet delivery ratio. However, the retransmission strategy of SW-MER uses a predefined number of packets for retransmission regardless of the channel condition. Although this strategy has been shown to improve the packet delivery ratio, there may be occasions where the protocol might be sending unnecessary copies of the same packet. The protocol included in this paper addresses this particular problem including an adaptive retransmission strategy that changes the number of copies to be retransmitted according to the quality of the channel. Our performance evaluation demonstrate that, compared with the SW-MER protocol, this new retransmission strategy can increase the throughput of the protocol by up to 15% without compromising its packet delivery ratio.

The remainder of the paper is organized as follows. Section II includes a description of some traditional logical control protocols used in underwater communications. Section III describes the proposed adaptive protocol. Section IV describes the Bernoulli error model, the underwater channel trace, and the two-state Markov error model. Section V includes the performance evaluation of the adaptive protocol. Finally, Section VI concludes the paper and presents directions for future research.

II. RELATED WORK

The half duplex nature of acoustic communication transducers has limited the design of logical link layer protocols for

¹Professor on leave from Universidad del Norte, Barranquilla, Colombia. www.uninorte.edu.co.

underwater communication to the well-known Stop and Wait protocol introduced in [2], or some variants of it such as the ones introduced in [3], [4], and [5]. In the original version described in [2], the transmitter sends one packet and waits for the corresponding acknowledgment before transmitting the next packet. If the sender does not receive the acknowledgment (meaning that the packet either arrived with errors or did not arrive at all) within certain time (retransmission timeout timer or RTO), it transmits the packet again. Packets are retransmitted as many times as necessary or until the maximum number of retries is reached. It is well-known that the Stop and Wait protocol not only provides worse performance with increases in the propagation delay but deteriorates even more in high error prone channels.

In the Stop and Wait variant proposed in [3], the sender transmits new packets one at a time. In case of a missing packet or a packet in error, the protocol retransmits a window of packets containing i copies of the missing packet. The process is repeated until the maximum number of trials is reached or the acknowledgment is received. This mechanism increases the reliability of the protocol at the expense of a poorer channel utilization.

The variant proposed in [4] was the first Stop and Wait protocol that sent a window of packets per transmission opportunity. The sender transmits a window of m packets and waits for an acknowledgment packet containing information about all m packets, i.e. one ACK per window. After receiving the acknowledgment, the transmitter sends a new window of m packets that includes those packets that arrived with errors and new packets if there is space available in the new window. However, it only retransmits each packet in error once. This protocol provides better performance than the one proposed in [3], but requires more buffer space at the receiver to guarantee the ordered delivery of packets to the higher layer. Once the buffer is full, packets are dropped.

The Stop and Wait version described in [5] works similar to the one proposed in [4]. After the acknowledgment packet is received, if there were packets with errors, the transmitter sends a new window of packets but only with retransmitted packets. After these packets arrive without errors, a window of m new packets is sent. This protocol has the advantage of requiring less buffer space at the receiver than [4], improves the packet delivery ratio but presents lower throughput in high bit error rate scenarios.

In [1] a logical link layer protocol was proposed to improve the throughput of the network and the packet delivery ratio in underwater communications sending a window of packets per transmission opportunity and including an exponential number of copies of retransmitted packets when errors occur. Every time a window of packets is received, the receiver verifies which of them arrived with errors, and then it sends an acknowledgment vector that provides feedback about the reception of each of the packets in the window. Once the transmitter receives the acknowledgment, it builds another window of packets that includes retransmissions of the packets received in error plus new packets if there is space available. In

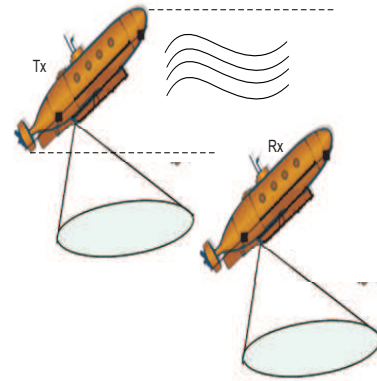


Fig. 1. Transmission between nodes located at different depths.

order to guarantee packet delivery even in the noisy underwater channel, the protocol retransmits the packets received in error a number of times per window that increases exponentially by two every time the same is received in error.

Although it has been shown that the SW-MER protocol improves the throughput and the packet delivery ratio, it sends a fixed number of copies regardless of the channel condition, which may lead to sending an unnecessary amount of copies of retransmitted packets in channels with high error variability, thus calling for an adaptive version.

III. PROPOSED PROTOCOL

In this section, the adaptive logical link layer protocol is described, including the processes to obtain and feed back the quality of the channel and the response taken by the sender.

A. Determining the Channel Quality

Depending on the distance between transmitter and receiver, the channel conditions as seen by each one might be different. Further, if the nodes move, these conditions will change with time. Figure 1 shows the scenario where two underwater unmanned vehicles explore the ocean sea at different depths.

When the transmitting UAV sends information to the adjacent UAV, it does not know about the channel conditions presented at the receiver. If the channel presents errors, those will be reflected in the data packets when they arrive at the receiver. The receiver then creates a vector of acknowledgments indicating which packets were received with and without errors, and sends it to the transmitter.

Saving information about the last transmission is not enough to predict what the channel condition would be the next time. Increasing the number of copies exponentially all the time might not be necessary, and a better throughput could be achieved if that number changed according to the channel quality. Based on this information, the four-state machine shown in Figure 2 was designed to determine the number of copies to be sent per incorrect packet, which considers the status of the channel during the last two transmissions (history) plus the current packet. The entire process works as follows:

- If the receiver received the last two packets correctly and the current one is also correct, the transmitter determines

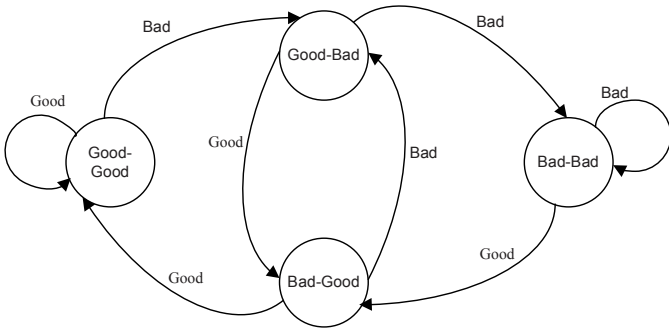


Fig. 2. Four-state machine representation.

that the channel is in the Good-Good state. In this case, the sender will decrease the number of copies per retransmitted packet by half, i.e., $c = c \div 2$.

- If the receiver received the last two packets correctly and the current one is bad, the transmitter determines that the channel is in the Good-Bad state. In this case, the sender will increase the number of copies per retransmitted packet by two, i.e., $c = c \times 2$.
- If the last two packets were received good and bad, and the current one is good, the transmitter determines that the channel is in the Bad-Good state. In this case, the sender will decrease the number of copies per retransmitted packet by half, i.e., $c = c \div 2$.
- If the last two packets were received bad and good, and the current one is bad, the transmitter determines that the channel is in the Good-Bad state. In this case, the sender will increase the number of copies per retransmitted packet by two, i.e., $c = c \times 2$.
- If the last two packets were received bad and good, and the current one is good, the transmitter determines that the channel is in the Good-Good state. In this case, the sender will decrease the number of copies per retransmitted packet by half, i.e., $c = c \div 2$.
- If the last two packets were received good and bad, and the current one is bad, the transmitter determines that the channel is in the Bad-Bad state. In this case, the sender will increase the number of copies per retransmitted packet by two, i.e., $c = c \times 2$.
- If the receiver received the last two packets incorrectly and the current one is also incorrect, the transmitter determines that the channel is in the Bad-Bad state. In this case, the sender will increase the number of copies per retransmitted packet by two, i.e., $c = c \times 2$.
- If the receiver received the last two packets incorrectly and the current one is good, the transmitter determines that the channel is in the Bad-Good state. In this case, the sender will decrease the number of copies per retransmitted packet by half, i.e., $c = c \div 2$.

Figure 3 illustrates the communication process between sender and receiver that conveys the channel quality information.

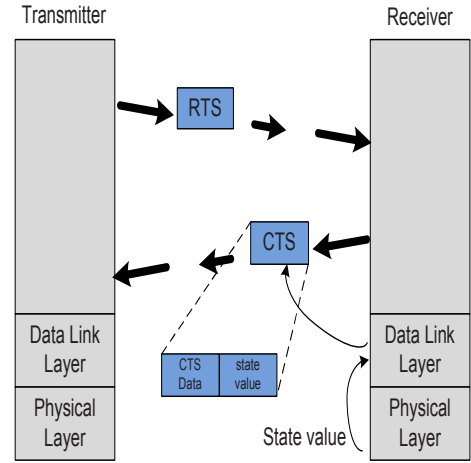


Fig. 3. Physical and data link layer interaction.

B. States of the Adaptive Logical Link Protocol

The adaptive SW-MER has the following states: Listen, verifying state of the channel, updating channel status history, updating copies of packets to be sent, waiting to receive packets, checking packets, enqueueing packets without errors, verifying packets to be sent to the upper layer, generating the ACK vector, and identifying packets sent with errors. Figures 4(a) and 4(b) show the sender and receiver state machines for the new protocol, respectively. The states and state transitions needed in the sender and receiver processes are explained next.

Listen: When a node decides to send packets to another node, it first selects the packets to be transmitted from the queue. Then, an RTS is sent with the total time needed to transmit a window of m packets and receive the acknowledgment vector.

Verifying the state of the channel: Once the RTS control packet arrives, the receiver saves the time that the transmission of m packets will take and prepares the CTS control packet. At this time the physical layer already had measured and identified the quality of the channel, and passed this information on to the data link layer. This information is then piggy backed in the CTS packet, which is finally sent to the transmitter.

Updating channel status history: Once the CTS packet is received, the transmitter extracts the channel quality information and uses the four-state machine presented in Figure 2 to determine the number of times packets in error need to be retransmitted in the next transmission opportunity.

Updating copies of packets to be sent: Based on the channel quality information and the four-state machine, the sender then assembles and transmits the new window of packets.

Waiting to receive packets: In this state, the receiver waits until the window of m packets arrives or a timeout occurs because at least one of the packets never arrived. The receiver identifies the missing packets using two information fields in the header of the packets, one that represents the position of the packet in the window and another one that represents the ordinal copy number.

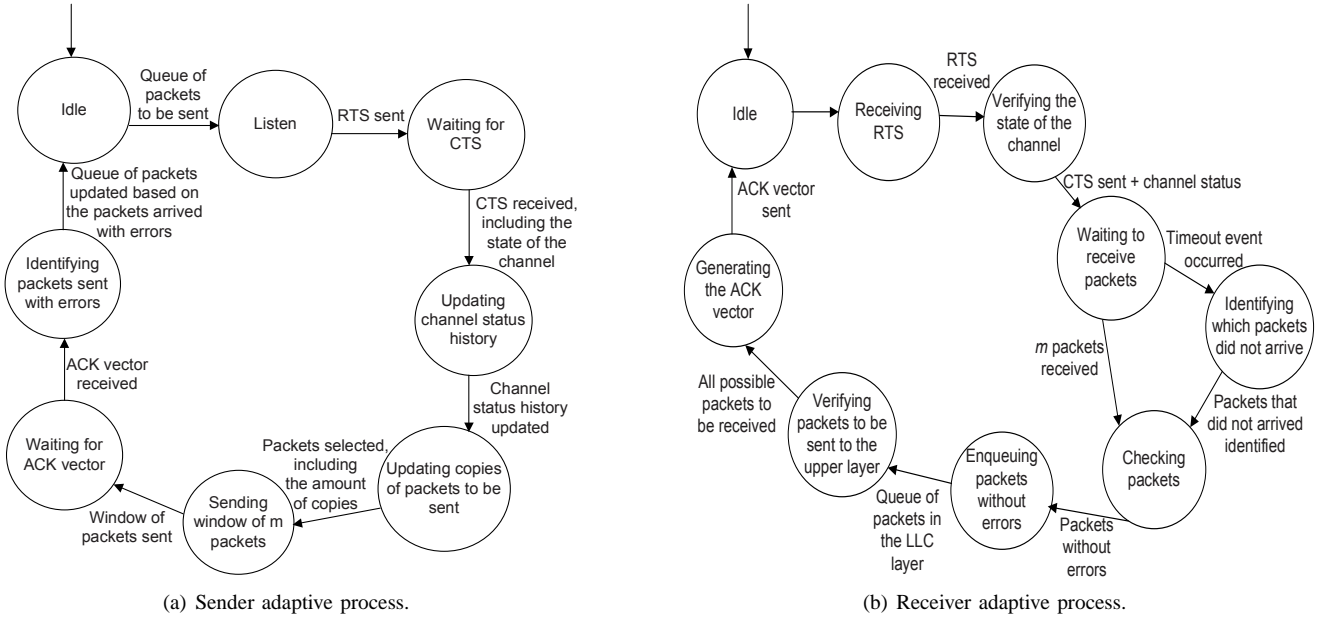


Fig. 4. Sender and receiver processes.

Checking packets: The receiver verifies if the packets arrived with or without errors by applying a well-known Cyclic Redundancy Check (CRC) function, and classifies them accordingly.

Enqueuing packets without errors: Packets that arrived without errors, are temporally enqueued in the buffer of the logical link layer. They stay there until a consecutive number of m packets arrive without errors. Those packets that were received incorrectly are eliminated. In the proposed protocol, it is very easy to calculate the buffer size B at the receiver to avoid packet drops, which is given by Equation 1 in which w is the size of the window in number of packets and t is the maximum number of trials that a data packet can be retransmitted. t is obtained from Equation 2

$$B = w \cdot t \quad (1)$$

$$t = \log_2(w) + 1 \quad (2)$$

Verifying packets to be sent to the upper layer: Once a consecutive number of m packets are received without errors, they are sent to the upper layer and deleted from the buffer. The purpose of sending the packets to the upper layer is to verify whether those packets belong to that node or they must be retransmitted to the next neighbor.

generating the ACK vector: In this state, the receiver is in charge of creating an ACK control packet to tell the transmitter the state of each packet received. This ACK packet is a vector in which each position is a one bit value that represents the status of the packet, where 0 means that the packet arrived correctly, and a 1 that the packets arrived with errors or did not arrive at all. At the end, the ACK vector is sent to the transmitter.

Identifying packets sent with errors: Once the ACK arrives to the sender, the vector is revised to identify which packets arrived with and without errors. Packets that arrived without errors are deleted from the sender's queue, the others are kept until a maximum number of trials is achieved or they are received correctly. The maximum number of trials is already defined in the Equation 2.

Figure 5 shows a graphical example of how the proposed protocol works using a window size of 6 packets. Let define a transmission process as an RTS/CTS/DATA/ACK transmission, assume that the channel was in a *Good – Good* state the last time, and a transmission process starts. Figure 5(a) displays the first transmission in which the channel status received in the CTS packet from the receiver indicates *Good*. Then, the state machine stays in *Good – Good* state, one copy of packets 1, 2, 3, 4, 5, 6 is sent and 2, 3 arrives with errors. The receiver sends back an acknowledgment informing this to the transmitter.

In the next transmission process, Figure 5(b), the transmitter sends two copies of the packets 2, 3 because it received a *Bad* channel status in the CTS packet from the receiver. Also, the status of the state machine changes to a *Good – Bad* state. As displayed in Figure 5(c), the channel status still is *Bad* at the receiver. As a consequence the state machine status goes to a *Bad – Bad* state and the amount of copies of each packet to be sent will exponentially increase by 2. Four copies of the packet 3 are sent due to arrived with error the last time. On the other hand, only two copies of packet 8 are sent because is a new packet.

Figure 5(d) shows the fourth transmission. The transmitter is informed that the channel status is *Good*, then the state machine state status is updated to *Bad – Good* and the amount of copies of each packet to sent is exponentially decreased by

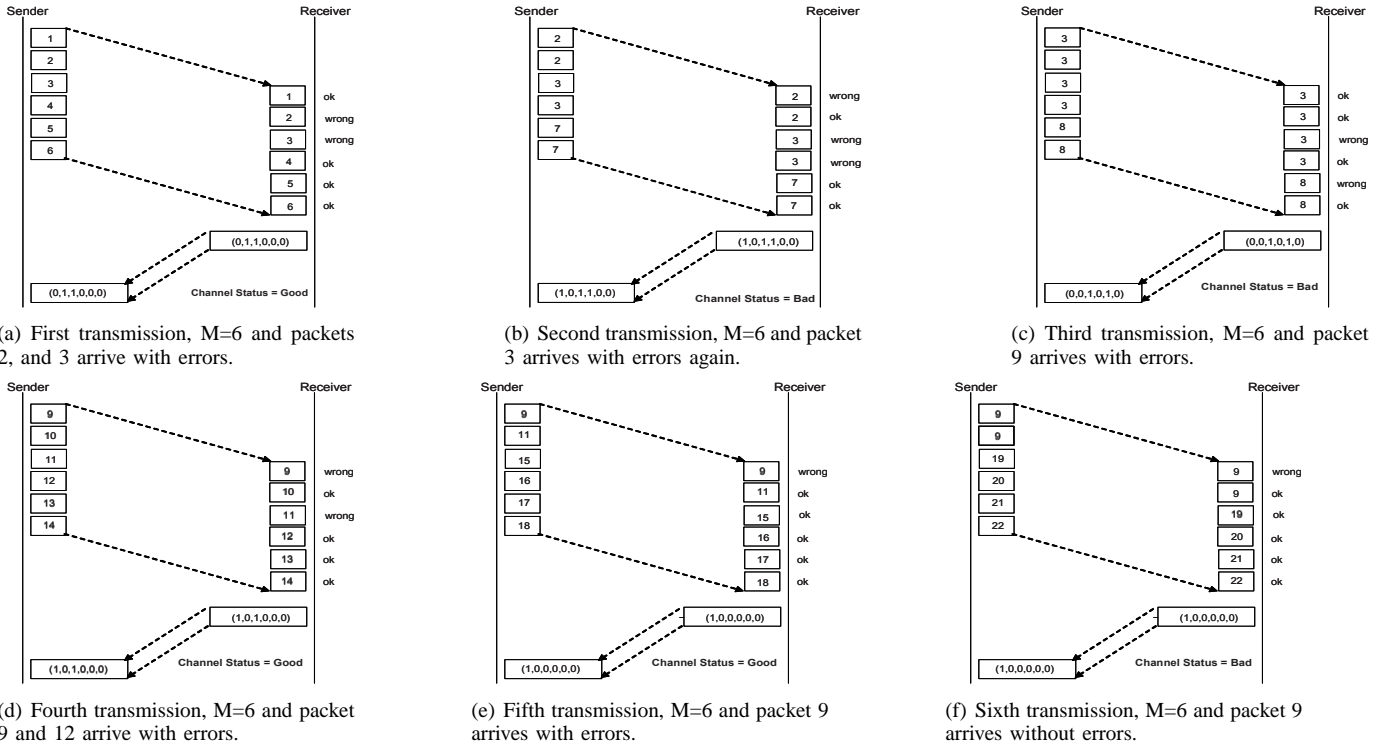


Fig. 5. Example of the proposed adaptive Stop and Wait sliding window-based mechanism.

2. Due to the transmitter is going to send new packets, only one copy from each one is sent. In Figure 5(e), only one copy of packets 9, 11 are sent although they arrived with errors the transmission before. This is because the last time the status of the channel state was *Good* – *Good* and still is *Good*.

IV. CHANNEL ERROR MODELS

To evaluate the performance of both protocols, two channel error models were considered. The first model corresponds to the simple Bernoulli model in which the Packet Error Probability (PER) is obtained with the following equation in which N represents the number of bits in the packet and BER is the Bit Error Rate:

$$PER = 1 - (1 - BER)^N \quad (3)$$

The implementation of the second model was based on similar approaches developed in [6] and [7] in which a two-state Markov chain is used to model errors in a wireless channel. For the channel characterization, results from [8] have been used, in which real measurements were taken in a shallow water using the parameters described in Table I.

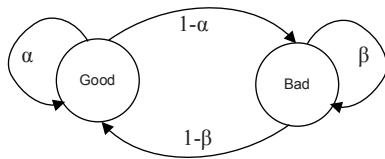


Fig. 6. Two-state Markov model representation.

TABLE I
CHITRE [8] PARAMETERS

Parameters	Values
sound speed	1550
mobile speed	0.7
range	50m

The channel impulses generated in From [8] were used to recreate the channel behavior (the amplitudes and the values) taking into account the Doppler effect when nodes are moving. Using Matlab simulations, a trace of 10 million impulses was generated using Orthogonal Frequency-Division Multiplexing (OFDM) and Binary Phase Shift Keying (BPSK) modulation schemes. The generated impulse results, +1, -1 and 0 were taken as a representation of bit with errors (+1 and -1 values) and bits without errors (0 value). With this trace, at the end the transition probability matrix A and the corresponding error probability matrix B of the two-state Markov chain that simulate the channel behavior are obtained. The Figure 6 represents the the two-state Markov chain of the channel error model. A and B are shown in 4, and 5.

Given initial values of A and B and the obtained trace, a Hidden Markov Model (HMM) approach using the Baum-Welch algorithm [9] was taken to generate the channel error model [10]. The initial values of the A and B matrices (A_0 and B_0) as well as the final values (steady state matrices) found by the model are as follows:

TABLE II
SIMULATION PARAMETERS

Error Model	Parameters	Values
Bernoulli	Speed of sound	1500 m/sec
	Data rate	2400 and 19600 b/s
	Range	50m
	Packets per window	8
	Packet size	150 and 300 bytes
Chitre [8]	Speed of sound	1550m/sec
	Data rate	2400 b/s
	Distance	50m
	Packets per window	8
	Packet size	150 and 300 bytes

$$A_0 = \begin{bmatrix} 0.98 & 0.02 \\ 0.05 & 0.95 \end{bmatrix} A_{ss} = \begin{bmatrix} 0.8116 & 0.1884 \\ 0.0095 & 0.9905 \end{bmatrix} \quad (4)$$

$$B_0 = \begin{bmatrix} 0.9 & 0.9 \\ 0.1 & 0.1 \end{bmatrix} B_{ss} = \begin{bmatrix} 0.9909 & 0.68 \\ 0.0091 & 0.32 \end{bmatrix} \quad (5)$$

V. PERFORMANCE EVALUATION

The performance evaluation of the proposed adaptive protocol is presented in this section. First, a general description of the parameters used in the simulations is presented. Then, the SW-MER and the adaptive SW-MER protocol are compared using the number of packet retransmissions and throughput as the main performance metrics.

A. Simulation Parameters

The simulation is carried out using the parameters included in Table II. Nodes with linear formations will be designed to evaluate the protocol. Two error models were considered, the Bernoulli model and a synthetic trace obtained from [8]. The values of distance between nodes, data rate, and other related parameters are presented in the table. For the Bernoulli model a bit error rate of 1×10^{-3} is used, control packet length's of 30 bits for RTS, CTS and ACK are applied in all the scenarios.

We assumed a linear topology with 5 UAVs, as shown in Figure 7. The traffic in the network was generated as follows. Each node generated flows to all other nodes sending packets according to a Poisson process with λ of 4 data packets per second (expected arrival in a time interval). The rate of the flows was set in such away that the load in the network was set to an specific desired level starting from 0 to 100%. In addition, a tagged flow sending packets from node *E* to node *D* was established and monitored. The simulations results of the performance in all graphs are related to the performance of that tagged flow. A simple network layer was included on top of the LLC layer to route incoming packets to the following adjacent node.

B. Throughput Evaluation

In this section the performance of both protocols is compared using throughput and number of copies per packet sent as the main performance metrics. This second metric is defined

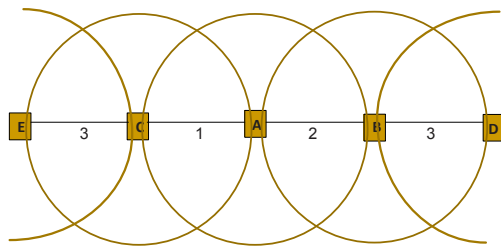


Fig. 7. Linear topology.

as the average number of copies that need to be transmitted per packet so it is finally received correctly.

Figures 8, 9 and 10 show the amount of copies per packet generated by the two protocols. The first two scenarios were executed with the Bernoulli model using a BER of 1×10^{-3} that generates a channel highly in errors. As it can be seen, holding the same packet size and increasing the load in the network cause that a lot of packets arrive with errors. Then the throughput of each protocol decreases as seen in Figures 11, and 12. The SW-MER protocol [1] reacts exponentially increasing the number of copies per packet each time a transmission process starts, to guarantee a successfully packet delivery. The protocol does not have into account that the channel is not in failure all the time like the proposed adaptive SW-MER. As expected, however the channel presents a lot of errors, the proposed protocol adapts to channel change behaviors generating less amount of copies to be sent, and improving the corresponding throughput, as seen in Figures 11, and 12.

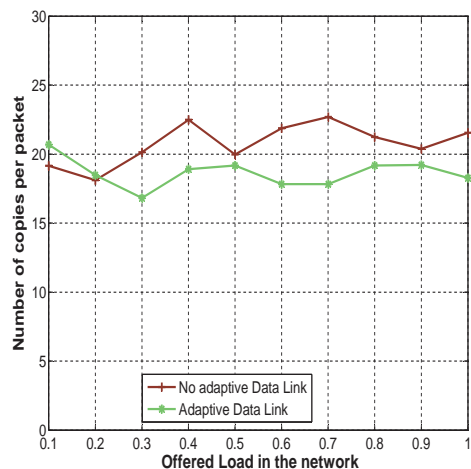


Fig. 8. Copies per packet using a Bernoulli error model, a window size of 8 packets, a packet size of 2400 bits, BER = 1×10^{-3} , and data rate of 2400 bps.

In the case of the underwater channel error model, similar experiments were performed to compare the results using the trace, Markov model, and parameters described in Section IV. As it can be seen from the third scenario displayed in Figures 10 and 15, the number of copies generated per packet with the protocol in [1] starts to be bigger than the amount the proposed adaptive SW-MER produces as soon as the load

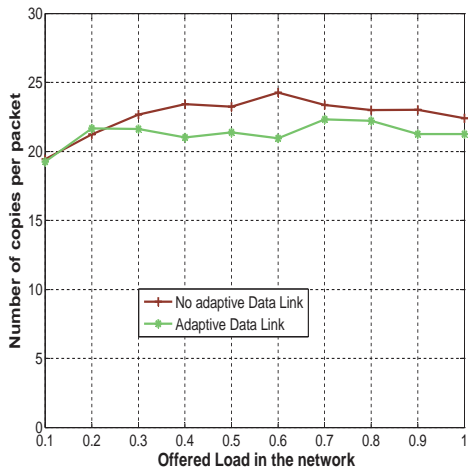


Fig. 9. Copies per packet using a Bernoulli error model, a window size of 8 packets, a packet size of 2400 bits, BER = 1×10^{-3} , and data rate of 19600 bps.

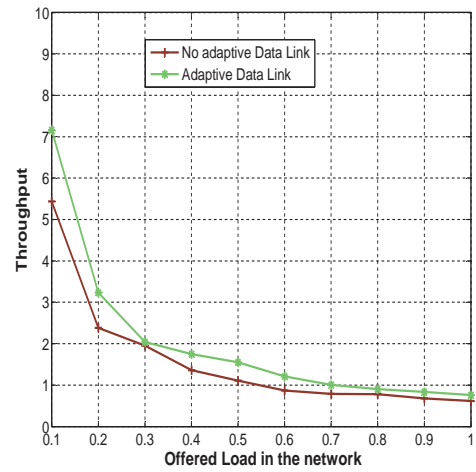


Fig. 11. Throughput using a Bernoulli error model, a window size of 8 packets, a packet size of 2400 bits, BER = 1×10^{-3} , and data rate of 2400 bps.

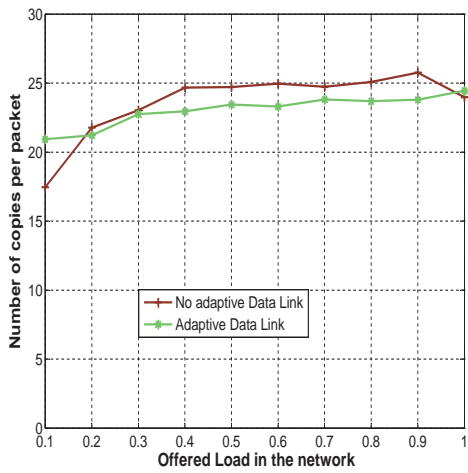


Fig. 10. Copies per packet using a shallow water error model, a window size of 8 packets, a packet size of 1200 bits, and data rate of 2400 bps.

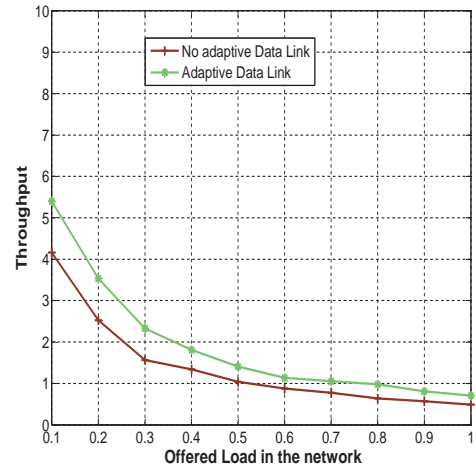


Fig. 12. Throughput using a Bernoulli error model, a window size of 8 packets, a packet size of 2400 bits, BER = 1×10^{-3} , and data rate of 19600 bps.

of the network is increased, affecting the throughput. As it is shown in Figure 15, the throughput starts to deteriorate but still the superiority of the proposed adaptive SW-MER is demonstrated. This is as a consequence of the reduction of the number of copies retransmitted in each window, incrementing the number of new packets to be sent.

From the results in the experiments can be seen that the adaptive SW-MER generates around 10% copies per packet less than SW-MER, especially when the network is highly loaded, and a throughput improvement in the order of 15% can be obtained, even using the synthetic trace that introduces a larger amount of errors. Other scenarios that can be evaluated are shown in Figures 13, 14 and 16. Figures 13 and 14 show that for window of 16 packets, however the throughput of both protocols are similar, the amount of packets generated for the previous protocol dramatically increases almost in a 50% compared with the amount produced by the proposed adaptive SW-MER.

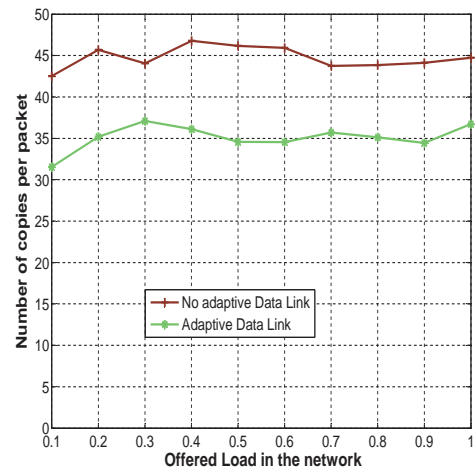


Fig. 13. Copies per packet using a shallow water error model, a window size of 16 packets, a packet size of 2400 bits, and data rate of 19600 bps.

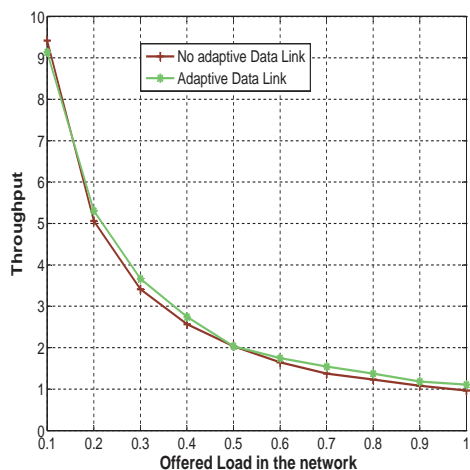


Fig. 14. Throughput using a Bernoulli error model, a window size of 16 packets, a packet size of 2400 bits, $BER = 1 \times 10^{-3}$, and data rate of 19600 bps.

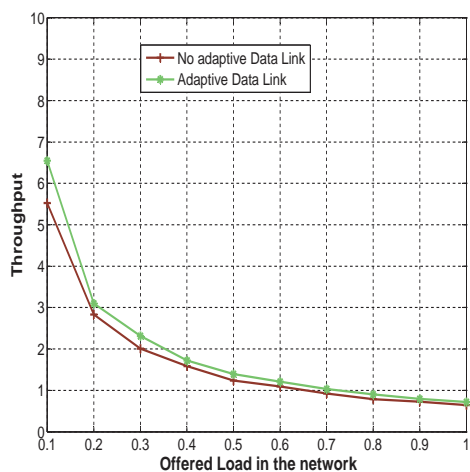


Fig. 15. Throughput using a shallow water error model, a window size of 8 packets, and a packet size of 1200 bits.

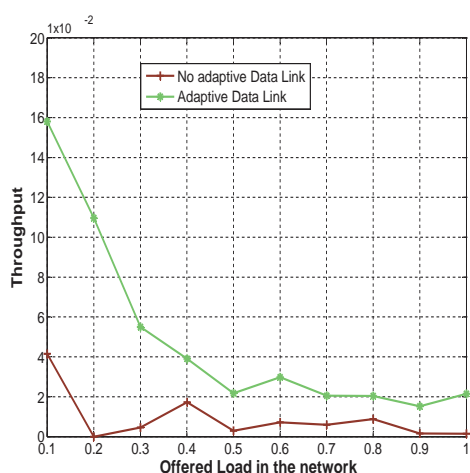


Fig. 16. Throughput using a shallow water error model, a window size of 8 packets, and a packet size of 2400 bits.

VI. CONCLUSIONS

In this paper, a new logical link layer protocol that adapts in response to underwater acoustic communication channel changes is presented. This adaptive protocol works considering past channel quality information to decide whether to increase exponentially or not the amount of copies per packet to be sent in following retransmissions, increasing the performance of the network. The adaptive protocol is compared with the SW-MER protocol described in [1] in terms of two metrics: the average number of copies per packet needed in order to successfully arrive at the receiver, and throughput. The simulation results for the first metric indicate that the adaptive SW-MER generates around 10% less copies. In terms of throughput, the simulation results show that the adaptive protocol offers a better performance, up to 15% better throughput, even in scenarios with very poor channel conditions. Although the superiority of the adaptive protocol is demonstrated, more research is needed in this area to improve the throughput even further.

ACKNOWLEDGMENT

This work has been partially supported by the National Science Foundation's Research Experiences for Undergraduates program under grant No. 0754537.

REFERENCES

- [1] D. Jabba and M. Labrador, "A Data Link Layer in Support of Swarming of Autonomous Underwater Vehicles," in *Proceedings of MTS/IEEE Oceans, Bremen, Germany*, May 2009.
- [2] M. Schwartz, *Telecommunication Networks*. Addison Wesley, 1998.
- [3] A. R. K. Sastry, "Improving Automatic Repeat-Request (ARQ) performance on Satellite Channels Under High Error Rate Conditions," *IEEE Transactions on Communications*, vol. 23, No. 4, pp. 436 – 439, 1975.
- [4] J. M. Morris, "Optimal Blocklengths for ARQ Error Control Schemes," *IEEE Transactions on Communications*, vol. 27, No. 2, pp. 488–493, 1979.
- [5] P. F. Turney, "An improved Stop-and-Wait ARQ Logic for Data Transmission in Mobile Radio Systems," *IEEE Transactions on Communications*, vol. 29, No. 1, pp. 68–71, 1981.
- [6] H. S. Wang and N. Moayeri, "Finite-State Markov Channels-A Useful Model for Radio Communications Channels," *IEEE Transactions on Communications*, vol. 43, pp. 163–171, February 1995.
- [7] W. Turin and R. V. Nobelen, "Hidden Markov Modeling of Flat Fading Channels," *IEEE Journal on Selected Areas in Communications*, vol. 16, No. 9, pp. 1809–1817, December 1998.
- [8] M. Chitre, J. Potter, and O. S. Heng, "Underwater acoustic channel characterization for medium-range shallow water communications," *OCEANS'04 MTS/IEEE Oceans, 2004*, pp. 40–45, November 2004.
- [9] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics*, vol. 41, pp. 164–171, February 1970.
- [10] L. R. Rabiner and B. H. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine*, pp. 4–16, 1986.