A Framework for Residual Energy Model in UnetStack Simulator for Underwater Sensor Networks

Sheeraz Ahmed
Faculty of Engineering and Technology,
Gomal University,
Dera Ismail Khan, Pakistan
asheeraz_pk@hotmail.com

Iman
Student of Computer Science,
Iqra National University,
Peshawar, Pakistan
Imaankhan850@gmail.com

Iman
Student of Computer Science,
Iqra National University,
Peshawar, Pakistan
Imaankhan850@gmail.com

Iman
Student of Computer Science,
Iqra National University,
Peshawar, Pakistan
Imaankhan850@gmail.com

Iman
Student of Computer Science,
Iqra National University,
Peshawar, Pakistan
Imaankhan850@gmail.com

Iman
Student of Computer Science,
Iqra National University,
Peshawar, Pakistan
Imaankhan850@gmail.com

***Abstract*-In recent years, underwater acoustic sensor networks (UASN) have attracted researchers' attention due to their various applications. UASNs faces some problems and challenges, such as limited bandwidth, high propagation delay, 3D topology, media access control, routing, resource usage and energy constraints. Unlike the Terrestrial Wireless Sensor Network (TWSNs) node, UASNs is plagued by energy constraints, which seriously affects the longevity and speed of the network. The simulation of UASN is common to researchers because it helps to analyze the function and performance of UASN before implementing and deploying UASN, which involves a lot of cost and time. Among the different simulation platforms that can be used to simulate UASN, UnetStack is one of them. It is an effective and well-known tool that can be used to simulate UASN, and has obvious advantages. However, the current UnetStack does not provide a direct function to monitor the energy of the node during the simulation process, which is very important. This article describes the design of residues and the framework for implementing the energy model in UnetStack. In addition, because of the experimental simulation, it can display the number of frames sent and received and the energy consumption of the node over time. In addition, the implemented energy model framework allows researchers to design energy-efficient routing protocols and load balancing methods.**

1. ***Keywords: Energy Model Framework, Network Lifetime, Energy Depletion, UnetStack.***

I. Introduction

As the demand for the development of marine resources increases, people are becoming more aware of the application of underwater sensor technology to marine monitoring. Underwater Acoustic Sensor Network (UASN) has been widely proposed as a promising solution to support various marine applications, such as pollution monitoring, coastal exploration, navigation assistance and mine identification [1].UASN faces some problems and challenges, such as limited bandwidth, high propagation delay, 3D topology, media access control, routing, resource usage and energy constraints [2].A typical UASN consists of multiple sensor nodes fixed on the seabed and wirelessly interconnected with one or more underwater gateways. Underwater sidewalks are specific nodes equipped with vertical and horizontal transceivers. The vertical transceiver is used to send commands and configuration data to the sensor nodes, and to further obtain the collected data. The horizontal transceiver is used to relay the monitored data to the surface station [3].The data is usually linked from the bottom to the surface station via a multi-hop path within the sensor network. Lifetime is one of the key factors in any communication network that uses battery-powered equipment. The longer the service life, the better the system. Compared with traditional terrestrial wireless sensor networks (TWSN), UASN presents severe challenges in terms of personnel and budget to manually extend the life of

the network. Unlike TWSN, UASN nodes consume energy for various reasons. Some of the reasons for the shortened network life are the environment, the high energy consumption of sensor hardware, and improperly designed protocols. Therefore, compared with WSN, UASN nodes require more energy, because the distance covered by the acoustic signal is very long, and more complex signal technology is implemented. In addition, multipath routing in UASNs affects the energy of the node when one node is involved in the transmission of data from another node. In UASN, nodes are not static like TWSN, but nodes move
Due to the different activities and environments of the underwater environment, it is usually in an underwater water flow of 2-3 m/sec [2]. With the exception of surface nodes (wells), most nodes deployed in UASN are subject to energy constraints and cannot be charged. However, it is not possible to use solar energy to charge or periodically replace discharged batteries in an underwater environment [1]. Simulation of UASN is a common aspect of research because it facilitates a cost-effective and time-consuming method of analyzing the operation and performance of UASN before implementing and deploying UASN. There are several simulation platforms that can be used to simulate UASN, but not all are open source. Compared with Aqua-Sim, DESERT and SUNSET, UnetStack supports a seamless transition from simulation to actual field deployment without changing code and design. Therefore, the compiled binary emulation code can be directly ported to any modem compatible with UnetStack for field or laboratory testing without additional cross-compilation [4]. In terms of efficiency, UnetStack supports real-time operating modes with discrete events. Therefore, UnetStack has become an ideal choice for researchers to conduct UASN simulation and then transition to actual field deployment. UASN's MATLAB and NS-2 simulations are also very popular in the literature. For MATLAB-based simulations, most are specific applications. Although MATLAB can be used for in-depth simulation, it is not possible to define personalized topology and power models. In addition, there is no defined definition method to monitor factors such as packet transmission, loss and collision that may be of interest to researchers or may even affect the performance of the submarine network. In addition, in MATLAB simulation, not all routing protocols are supported. Another important issue to consider is node mobility, and there is no ability to simulate node mobility in this simulator [5]. The latest version available is NS-2.36. Except for NS-2.30, NS2 does not provide any integrated software package that supports UASN simulation. The other simulation scripts NS-2.30 written in the above version are difficult to execute. Therefore, in NS-2.35 and later, UASN simulation needs to model all the characteristics and propagation models of underwater channels [6]. It uses C++ and Python to write scripts easily and supports visualization [7]. The UAN model has four main components: channel, PHY, MAC and underwater autonomous driving (AUV) model. The framework aims to simulate the behavior of AUV. The communication stack associated with the AUV can be modified according to simulation requirements. Generally, the default underwater stack includes a half-duplex acoustic modem, Aloha MAC protocol, and a common physical layer.

II. RELATED WORK

This part of the paper starts from various aspects of underwater communication research and introduces the importance of implementing an energy model in UnetStack. In addition, the energy model needs to be implemented in the UnetStack platform. Mandar Chitre et al. [8], UnetStack developers introduced a detailed introduction and overview of the UnetStack architecture. In addition, the author provides detailed information about the different services available and a set of predefined agents that provide the service in his contribution.

2.1 SIGNIFICANCE OF ENERGY MODEL IN UNDERWATER RESEARCH

As described in Section 1, node energy monitoring in UASN is very important in all aspects of underwater research, such as energy consumption analysis, design of routing protocols based on the following aspects, data collection strategies, and mitigation of energy gaps and load balancing. This section describes the importance of energy monitoring in the above research issues. Han Guangjie, etc. [1] A reverse routing protocol based on asymmetric link (AREP) is proposed to ensure bidirectional data communication between the source node and the destination node. The author discusses the influence of the directional beam width of submarine nodes on the communication link. In addition, the author conducted three case studies on communication links. These case studies show that for a directional antenna with a fixed beam width, the change in the relative position of two geographically adjacent nodes is likely to produce an asymmetric link. Compare AREP performance with adaptive feedback based on link state (LAFR) routing. AREP can shorten the transmission time and increase the data packet transmission speed in the underwater environment. Mari Carmen Domingo and Rui Prior [9] proposed a mathematical analysis of the total energy consumption of underwater acoustic networks in shallow and deep water. Relaying or grouping based on routing protocols for shallow and deep water scenarios has been studied. Finally, the author concludes that the routing protocol based on clustering scheme can save more energy and show better performance in shallow water. Xing Guanglin others. [10] The UASN was deployed on a named data network (NDN), and the NDN-based topology explored the power consumption of the NDN-based UASN under shallow and deep water conditions. Relay network. Simulations were carried out in NS-3 and MATLAB to analyze the results of the energy consumption model of NDN-based UASN relays in shallow and deep water.

MATLAB does not provide any methods for defining custom topologies or methods for monitoring factors such as packet transmission, collision, and loss. In the first step, the sensor nodes are deployed in a rigid graph-based topology, and the physical data is relayed to the short-distance data collector through multi-hop acoustic communication. Then, in the second stage, the AUV periodically accesses the data collector to recover the data with high-speed communication in visible light. Simulations were conducted in MATLAB 2016b, and the results show that the topology optimization scheme can extend the life of the network. The author proposes an algorithm that can overcome the interference during the transmission of data packets by defining a unique data packet retention time for each sensor node. The variable transmission range of the sensor node is used to perform position less band gap attenuation. The variable transmission range of the sensor node is used to perform band gap attenuation without position. Although these results prove the performance of the proposed method, they cannot be used for real-time implementation. Therefore, the results obtained using UnetStack can be used not only for simulation, but also for real-time deployment. Therefore, the UnetStack energy model implemented in this paper proves its necessity.

2.2. NEED FOR ENERGY MODEL IMPLEMENTATION IN UNETSTACK

UnetStack, Aqua-Sim, SUNSET and DESERT are some of these tools, which can be used actively and downloaded for free. Based on the most popular NS-2 simulator, Aqua-Sim [13] is a package-level simulation platform. Similar to NS-2, Aqua-Sim also uses an object-oriented design style and provides researchers with a wealth of underwater protocols. However, currently in Aqua-Sim, it does not support a seamless transition from simulation to field deployment, because it only focuses on simulation and simulation. DESERT [14] and SUNSET [15] are simulation, simulation and experiment tools based on NS-2 and NS2-Miracle. Similar to DESERT, SUNSET can easily handle simulation, seamless transition between simulation and field test. In terms of efficiency, DESERT and SUNSET are extended from NS-2. For example, both platforms have a main single-threaded process, and events are scheduled to run in sequence by a strict event scheduler, which is sensitive to events of limited duration. On the other hand, UnetStack supports real-time operation mode with discrete events. Therefore, the compiled binary simulation code can be directly ported to any modem compatible with UnetStack (such as Subnero) for field or laboratory testing without additional cross-compilation [4]. In addition, in the case of a transparent transition from simulation to simulation or field deployment, SUNSET and DESERT are based on Ns-2 and Ns2-Miracle, which are mainly Ns-2 simulators' caution events. Therefore, during the transition from discrete event-based simulation to real-time simulation distributed on these platforms, major changes to the code and design may be required, which may cause other problems. For example, when simulating an application, if it uses centralized global network information, you must be very careful when transferring code from the simulation to the application when simulating the same application. For example, it is difficult to identify problems related to synchronization of events in the simulation. For example, it is difficult to identify problems related to synchronization of events in the simulation. In addition, in DESERT, the packet conversion method is not very practical, which may lead to higher packet conversion overhead [4]. UnetStack uses an agent-based architecture and supports real-time simulation. Therefore, the same compiled binary code used in the simulation can be directly transplanted to the underwater modem compatible with UnetStack without cross-compilation for simulation or field test. Therefore, UnetStack has become an ideal choice for researchers to conduct experiments. Therefore, it is very important to implement an energy model that is not currently available in UnetStack.

III. RESIDUAL ENERGY MODEL FOR UWSN

The main purpose of the proposed work is to provide a residual energy model framework to deepen the energy-related algorithms in UnetStack. In this proposed work, the author Guangjie Hana et al. proposed a residual energy module. [1] Implemented in UnetStack. The detailed implementation of the existing energy model framework described in Section 4 below can be expanded modified according to the specific needs of researchers or industries. In this proposed work, the initial energy of the node is derived for each transmission and reception of the data packet. As described by the author Guangjie Hana et al. [1] Equation. (4) Represents the energy consumed during this period to transmit and receive m-bit data packets.

3.1 ENERGY CONSUMPTION DURING THE TRANSMISSION OF M-BIT PACKET

The following equation (1) indicates energy consumption during transmission of m-bit packets.

$$E_{tx}(m, l) = m * E_{elec} + m * T_b * C * H * l * e^{\#(f) * l} \quad (1)$$

Where,

E_{elec}–the energy consumed by the transmitter electronic to process one bit of data

l - The transmission distance

T_b- bit duration

H - Water depth

• C - an empirical constant calculated using Equation. (2)

$$C = 2_{\pi} * 0{:}67 * 10^{-9:5} \quad (2)$$

•$^{\#}$ (f) - a frequency dependent medium absorption coefficient (in db/km) calculated using Equation. (3)

$$(f) = 0{:}036 * f^{3/2} \quad (3)$$

Where, *f* is frequency of sound wave (in kHz) underwater.

3.2 Energy consumption during the reception of m-bit packet

The following Equation (4) indicates the energy consumption during the reception of m-bit packet.

$$E_{rx}(m, l) = m * E_{elec} \quad (4)$$

IV. IMPLEMENTATION OF ENERGY MODEL IN UNESTACK

This part of the article introduces the implementation of the residual energy model framework in UnetStack (see section 3). First, an overview of UnetStack is introduced, followed by a physical proxy and detailed information about the analog modem used in UWSN simulation. Finally, the detailed implementation of the residual energy model framework is given.

4.1 Overview of UnetStack

UnetStack is part of the Unet project and was developed at the National University of Singapore's Acoustic Research Laboratory in 2004. UnetStack is a proxy-based stack, which is the foundation of the submarine network simulator and can be easily used for the deployment and testing of submarine networks. In the default stack UnetStack, a collection of software agents representing different layers of the network stack is provided. These agents provide well-defined services from all layers of the network stack.

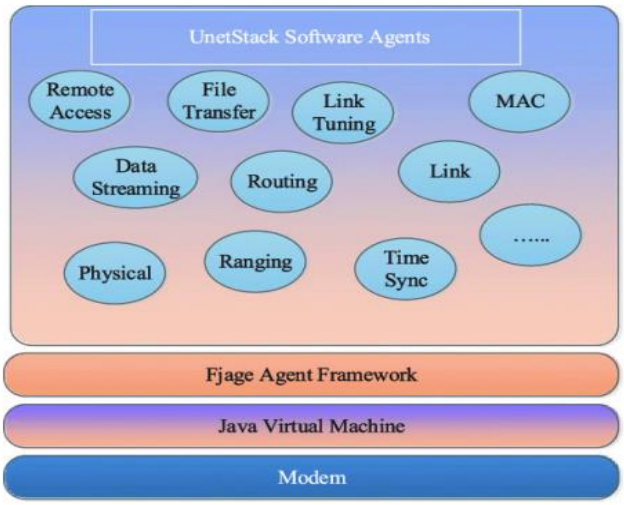

Fig 1: UnetStack architecture

This agent-based approach has produced a flexible network stack, and its solution is based on multiple layers, so that software-defined submarine networks can be quickly designed, simulated, tested, and deployed. In addition, because the stack is extensible, new agents and services can be added or existing agents can be replaced to meet development requirements. As shown in Figure 1, UnetStack's architecture uses a service-oriented architecture approach, where the stack defines a set of software agents that provide well-defined services. The agent plays the same role as each layer of the traditional network stack. To achieve this agent-based approach, UnetStack uses the open source fjage lightweight agent framework (Java and Groovy agent framework) 5. The fjage framework provides the basic implementation that forms the basis of UnetStack. In addition, UnetStack uses fjage to define agents and their services in its stack. Agents are the basis of UnetStack, exchanging messages, providing services and implementing protocols. We can also develop our own agents. UnetStack provides the UnetAgent base class for this, which implements most of the basic behaviors required by well-behaved agents. The agent in UnetStack is an independent software component that provides well-defined functions and more flexible interaction with other agents. Agents interact with each other through messages. The types of messages used in UnetStack can be divided into request, reply and notification. There is always an associated request in response, and no notification is requested. The message does not always have to be sent to a specific agent, but it can also be published on the topic. Any agent subscribing to this topic will receive broadcast messages about this topic. Unsolicited notifications are usually sent on topics associated with the agent; because the agent does not know in advance which other agent is interested in the notification. A well-integrated set of fully integrated requests, responses and notifications are called services. If the agent provides the service, it will announce the service by saving it to the "directory". Services can define functions, and these functions represent optional functions that the service can choose to implement. Agents release this function so that other agents can interrogate [8]. The main services defined by UnetStack are physical services, datagram services, MAC services, routing and routing maintenance services, transmission services, remote access services, telemetry services, the service links and node information services [8].

4.2 Residual energy model in UnetStack

In the implementation of energy models, the HalfDuplexModem class is extended by implementing a class called EnergyModelModem, which adds energy monitoring functions. The hierarchy of extension classes is shown in Figure 2 and Algo. Figure 3 shows the implementation implemented in the extended EnergyModelModem class. The extended EnergyModelModem class should be able to register to handle incoming datagram requests and notify it of physical and datagram services. In addition, the expansion modem must monitor each transmission and reception to calculate the energy consumption of the node. To this end, the

extended modem agent must monitor two physical agent notification messages, which are the frame transmission notification and the received frame notification of the base class when processing the transmission and reception of node data separately Send these notification messages. To send these messages, the HalfDuplexModem class calls send() defined in the fjage agent and passes the message instance as a parameter, because HalfDuplexModem is an extension of the UnitAgent class, and UnitAgent is extended from the Agent class of fjage, from the top of the hierarchy Call "send()" in the Agent class.

Therefore, in the extended EnergyModelModem that reaches the last level of this hierarchy, this "send()" will be overwritten and defined to check whether the message instance received as a parameter is a TxFrameNtf or RxFrameNtf message instance. The energy consumed for this is if the parameter is For the instance of TxFrameNtfmessage, the data transmission

is calculated; if the parameter is an RxFrameNtf message, the received data is calculated, because the design of the energy model that we use to calculate the transmission energy needs to calculate the size of the transmission energy.

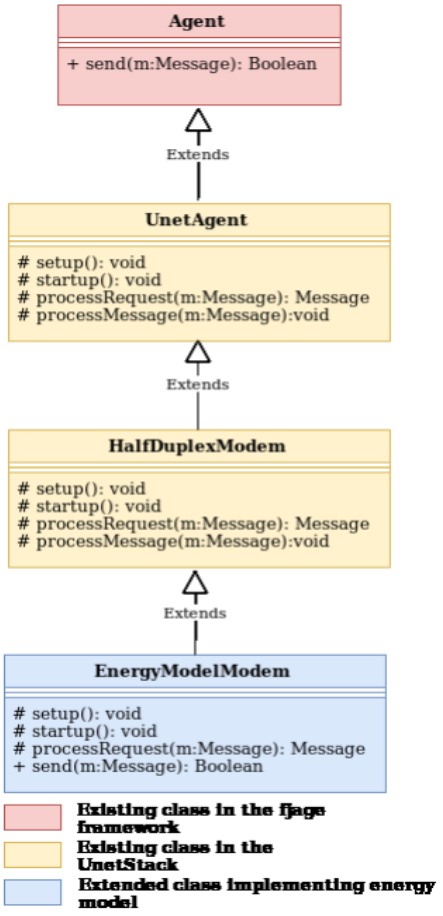

Fig 2: UnetStack: Energy model class diagram

Data distance between the sending and receiving nodes. We need to obtain the data sent from the node and the location of the node. These are the data transmission nodes retrieved by the datagram request message generated when processing the request.

Finally, in the simulation, the EnergyModelModem customized for monitoring the remaining energy was added by configuring the modem parameters, so it was added as a physical agent to the container for the nodes running in the simulation.

Table 1: Parameter for Energy Model

Parameter	**Value**
Channel Center Frequency (f)	10 KHz
Empirical Constant (C)	1.3312_e-9
Thorp's Constant (@(f))	$0.036*f^{1.5}$
Bit Duration (T_b)	0.001 sec

Table 1 lists the parameters used in the energy model implemented with reference to equations (1) and (4) in Section 3. Table 2 gives the other two parameters, water depth (H) and transmission distance (l), depending on the simulation.

Table 2: Simulation Parameter

Number of nodes	7
Communication range	1000 m
Simulation time	1 min., 1hr.
Initial energy	10 J
Water depth (H) (Node 1 through 7)	0, 800, 1500, 2200, 2800, 600, 2000
Transmission distance (l)	Euclidean distance between

V. RESULTS AND ANALYSIS

This section of the article introduces the simulation topology, parameters and different schemes for verifying/demonstrating the residual energy model implemented in UnetStack. In addition, the topology was simulated for duration of 1 minute. (The first case) 1 hour (Case 2). In addition, case I and case II were simulated, with or without receipt (ACK). The successful implementation of the remaining energy model in UnetStack is illustrated by parameters, such as the total number of data packets sent/sent and received by all active nodes and their relative energy consumption. Case-I provides in-depth simulation of the implemented modules, while Case-II provides the robustness of the implemented modules with detailed simulation. ACK hop-to-hop simulation ensures reliable

package delivery. In the UnetStack simulation, the RewableLink type uwlink agent available in the default stack provides link layer services with segmentation/reassembly and reliability at the link level. When using uwlink to send data, you can activate hop-by-hop ACK to achieve reliable transmission. This can be done by setting the reliability field to true. If the router agent is used to send data, uwlink is the default link agent. When the ACK skip function is available by default, or when adding a route, you can explicitly control it using the reliability field set to true or false. Router agents provide routing services based on routing tables.

5.1 SIMULATION TOPOLOGY AND PARAMETERS

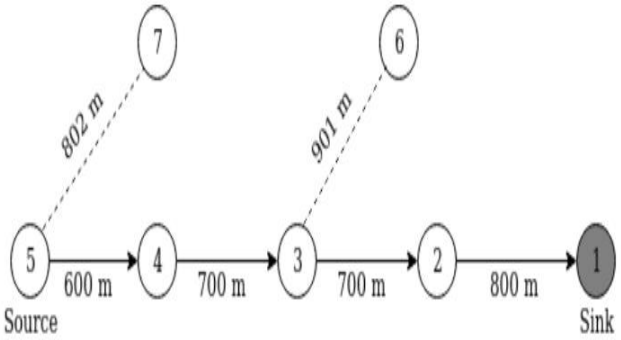

Fig 3: Simulation topology

This section introduces the simulation topology and its parameters as well as the values related to the residual energy model implemented in UnetStack. Figure 3 shows the simulation topology consisting of 7 nodes. As shown in Figure 3, circles represent nodes, and dotted lines connect nodes as neighbors to each other. In addition, the solid line with arrows indicates the data flow, and the value next to the line indicates the Euclidean distance (l) between the connected nodes. Table 2 lists the simulation parameters and the parameters configured for the energy models H and l with reference to equations (1) and (4) in Section 3.

5.2 CASE-I:

This section introduces topology simulation, as shown in Figure 2. The duration of 1 minute is 3, with and without ACK. In addition, node 5 and node 1 are configured as a source and a sink, respectively.

Total packets sent/forwarded: The multi-line diagram in Figure 4 illustrates the total number of packets sent/sent by all nodes with hop-by-hop ACKs defined in the simulation. Since Node-1 is the receiver, it only sends ACK packets. The data packet sent by node 4 includes the data packet to be sent to node 3 and the ACK is returned to node 5. Similarly, the data packets sent from nodes 3 and -2 include the data packets sent to nodes 2 and -1, respectively, and the returned ACK.

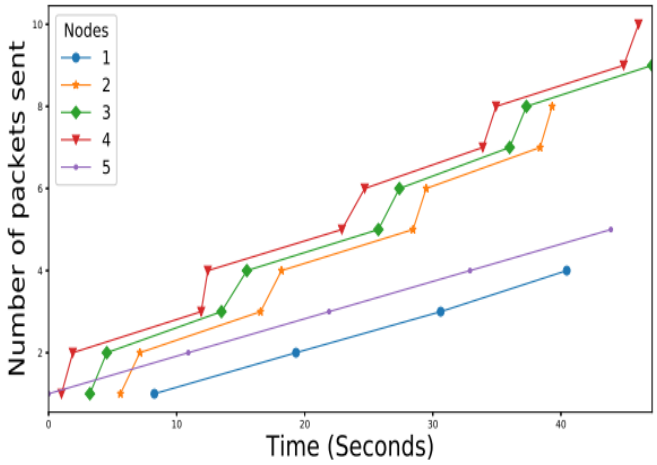

Fig 4: Total packets sent/forwarded with ACK

Eventually, Node-5, which is a leaf node, sent its own data packet. As shown in Figure 4, each mark on the graph represents an event sent by data packet transmission or by receiving confirmation, except for node 5, which only occurs in the sent data packet and node 1 ACK. In addition, as shown in Figure 4, node 5 sends 5 data packets,

Therefore, 10 sent events (5 packets + 5 ACKs) are recorded at nodes 4 and -3. On node 2, 9 transmission events (5 packets + 4 ACKs) were recorded, so 4 transmission ACKs were recorded on node 1. In addition, in Figure 4, you can also see the delay in sending packets and ACK sending events.

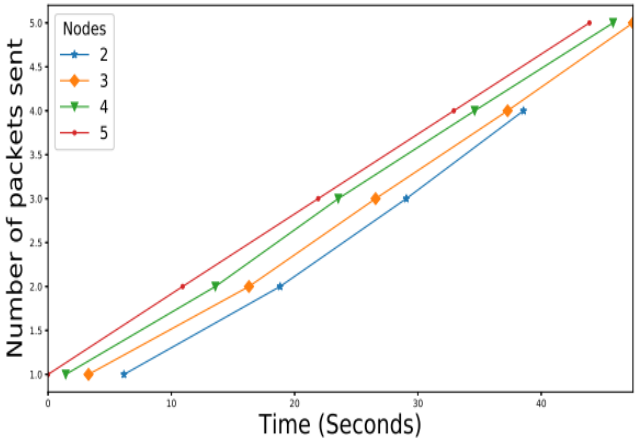

Fig 5: Total packets sent/forwarded without ACK

Figure 5 illustrates the total number of data packets sent by all nodes with no ACK defined in the simulation. We can notice that node 1 has no graph, because it is about the receiving node, and here it does not send ACK. The data packet sent by node 4 includes the data packet to be sent to node 3. Similarly, packets sent from nodes 3 and -2 include packets sent to nodes 2 and -1, respectively. Eventually,

Node-5, which is a leaf node, sent its own data packet. As shown in FIG. 5 without ACK, the number of data packets sent by node 5 corresponds to the number of data packets transmitted by intermediate nodes 4 and -3. At node 2, the number of retransmitted data packets is one less than the previous node due to the end of the simulation. The numerical values shown in Figures 4 and 5 prove that the energy model implemented in UnetStack is correct for the normal operation of packet transmission/retransmission events.

Total packets received: The multi-line diagram in Figure 6 illustrates the total number of packets received by all nodes with hop-by-hop ACK. Since node 1 is the receiver, it only receives data packets sent by node 2. Leaf node Node-5 only receives the ACK sent by node 4. The remaining nodes receive the previous hop and ACK sent from the next hop.

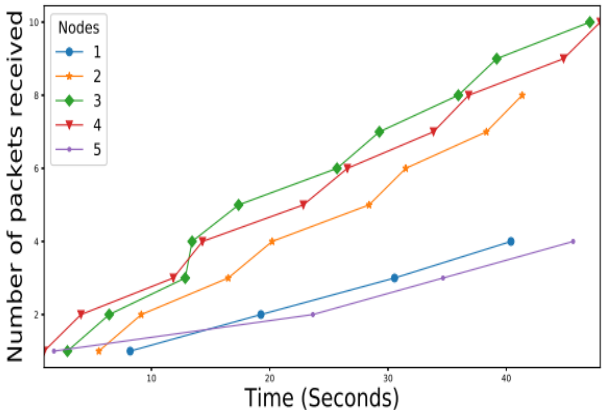

Fig 6: Total packets received with ACK

As shown in FIG. 6, each mark on the figure indicates that a packet or ACK event is received, except for node 5, which occurs only in the received ACK and the received packet of node-1. In addition, as shown in Figure 6, node 5 sent 5 packets, so nodes 4 and -3 recorded 10 reception events (5 packets + 5 ACKs). At node 2, 8 received events (4 packets + 4 ACKs) are recorded. Therefore, at node 1, 4 received packets are recorded. In addition, the data packets sent and received can be mapped in Figures 4 and 6, respectively. Figure 7 illustrates the total number of packets received by all unacknowledged nodes. It should be noted that since it is the source node and does not receive any ACK, there is no graph for node 5. Node 1 on the receiving side only receives data packets transmitted by node 2. The remaining nodes receive the data packet sent from the previous node. As shown in Figure 7 without ACK, the number of data packets received by node 1 corresponds to the number of data packets transmitted by intermediate node 2 (Figure 5). The same is true for Node-3 and -4.

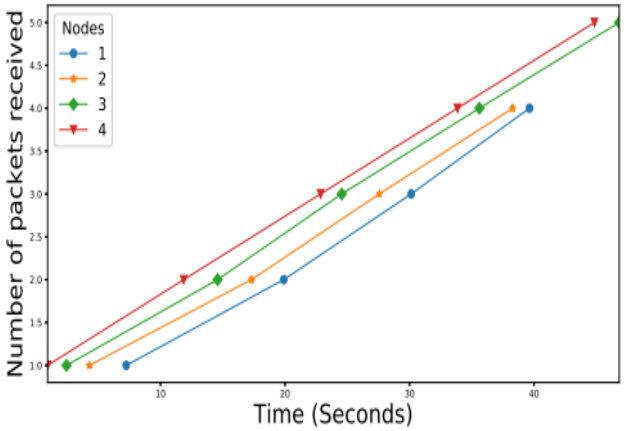

Fig 7: Total packets received without ACK

The values shown in Figures 6 and 7 demonstrate the correct function of the energy model implemented in UnetStack for the events received by the packet.

Depletion of node energy: This section describes the successful implementation of the main objectives of the proposed work.

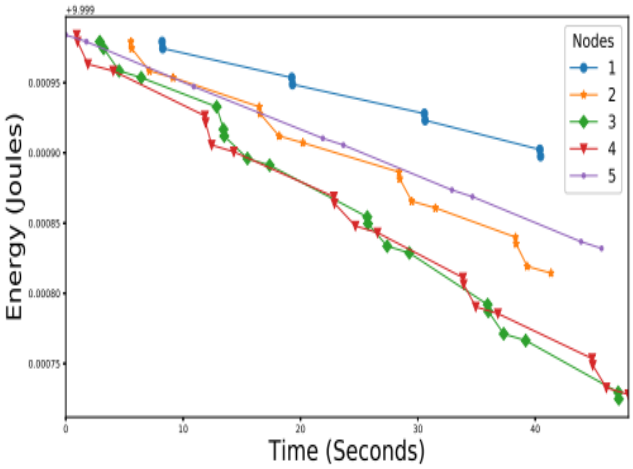

Fig 8: Energy depletion for all active nodes with ACK

The multi-line diagrams shown in Figures 8 and 9 represent the energy consumption of all active nodes, which are the result of all transmissions and receptions performed by the nodes. The figure does not show the energy consumption of nodes 6 and -7, because these nodes do not send or receive data packets (Figures 4, 5, 6 and 7), but can detect the transmission (Figure 3) the energy remains unchanged. The mark on the graph indicates that a send or receive event has occurred on the node. Figure 8 shows the energy consumption curve of ACK hop-to-hop nodes. We can see from the figure that node 1 (ie the receiver) consumes the least energy. Since node 1 does not participate in forwarding the data packet to the next hop, it receives the data packet from node 2 and returns the ACK of the received data

packet to node 2. Then, node 5 as a source has less energy consumption because it only sends the data packet to the next hop, node 4, and receives ACK from node 4. In addition, nodes -2, -3, and -4, which are intermediate nodes, have higher energy consumption because these nodes receive data packets from the previous hop, return ACKs from the previous hop, and transmit the data packets to the next hop and from the next One hop to receive ACK. As shown in FIG. 8, the number of marks on a single node graph is the sum of the number of marks on each node graph in FIGS. 4 and 6. In addition, it can be seen in FIG. 8 that energy consumption for packet transmission = forward packet> ACK transmission> packet reception> ACK reception.

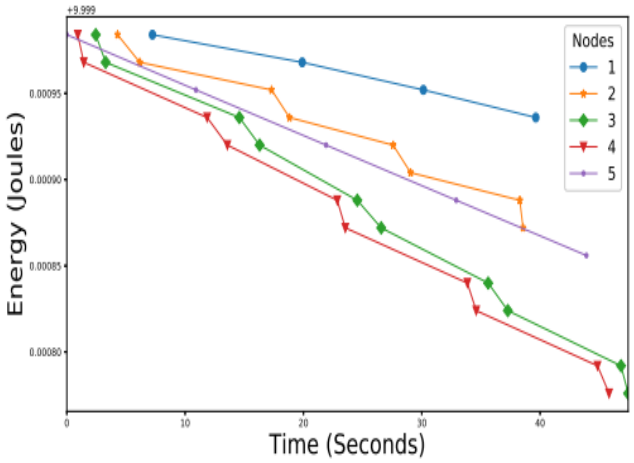

Fig 9: Energy depletion for all active nodes without ACK

The graph in Figure 9 shows the node energy consumption without ACK. It can be seen that, similar to the graph in Figure 8, node 1 has the lowest energy consumption because it only participates in receiving data packets. Node 5 has run out of energy and cannot send its own data packets. The other nodes lose the energy to receive the data packet in the previous hop and transmit the data packet to the next hop. Similar to FIG. 8, the total number of marks can be checked between FIG. 9 and FIGS. 5 and 7. The values shown in Figures 8 and 9 demonstrate the correct function of the energy model implemented in UnetStack during the transmission, reception, transmission, and ACK transmission and reception of data packets.

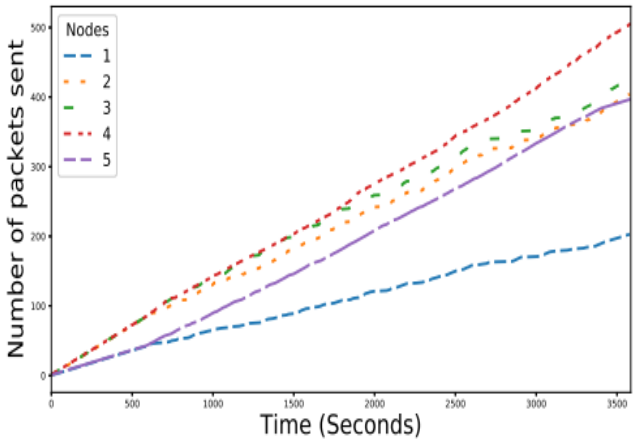

Fig 10: Total packets sent/forwarded with ACK

5.2 CASE-II: EXTENDED SIMULATION AND ANALYSIS

This section introduces extensive topology simulation, as shown in Figure 2. 3. The duration is 1 hour, with and without ACK. In addition, node 5 and node 1 are configured as a source and a sink, respectively. Compared with case I, in case II, the initial energy of the node is configured to 0.5J.

Total packets sent/forwarded***:*** The multi-line diagram in Figure 10 illustrates the total number of data packets sent by all nodes through a hop-by-hop ACK. Since node 1 is the receiver, it only sends ACKs for the packets received from node 2. The data packets sent from nodes 4, -3 and -2 include the data packet sent at its next hop and the acknowledgment sent at its previous hop. As shown in FIG. 10, since node 1 only sends an ACK, node 1 sends the least number of data packets, which depends on the number of data packets it receives from node 2. The number of data packets sent by node 5 is higher than that of node 1, but less than other nodes. Since node 5 must only send its own data packets, or retransmit some data packets when no ACK is received from node 4. However, for the intermediate nodes (nodes 2, -3 and -4), they have the same task of transmitting the packet at the next hop and sending the ACK to its previous hop. The number of packets sent is different because it depends on the number of packets they transmit or retransmit and the number of ACKs they send, which in turn depends on the number of packets they receive. . In this case, in FIG. 10, the node 4 has the maximum number of data packets sent.

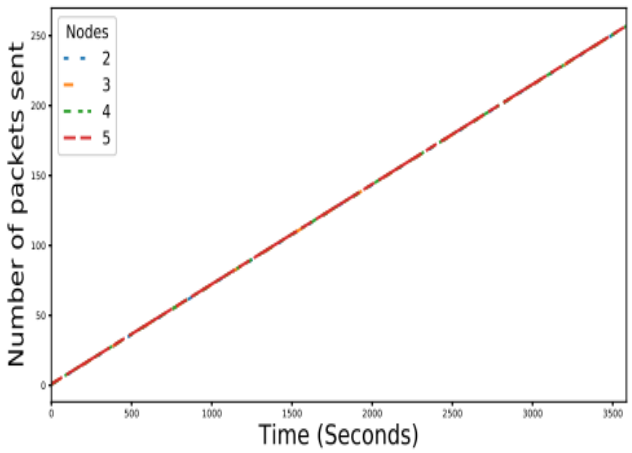

Fig 11: Total packets sent/forwarded without ACK

Figure 11 shows the total number of packets sent by all nodes without ACK. Here, in addition to node 1, sending data packets involves the source 5 nodes, and forwarding data packets involves the intermediate nodes (nodes 4, -3 and -2). Since all data packets sent from node 5 are transmitted through each intermediate node, the power consumption of all these nodes is the same. As shown in Figure 11, node-1 (that is, the receiving node) does not send any data packets. The number of data packets sent by intermediate nodes (nodes 4, -3 and -2) is the same as the data packets sent from node 5. Therefore, in FIG. 11, the graphs of these nodes overlap.

Total packets received:

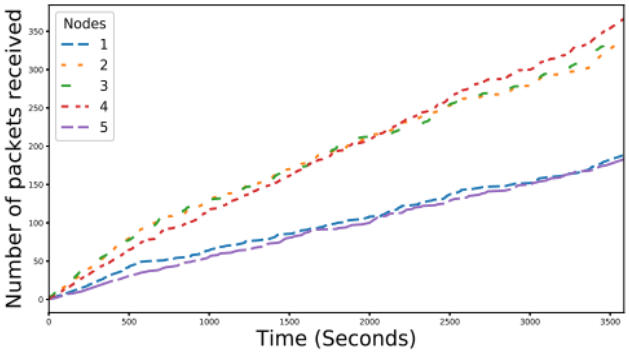

Fig 12: Total packets received with ACK

The multi-line diagram in Figure 12 illustrates the total number of packets received by all nodes with hop-by-hop ACK. Since node 1 is the receiver, it only receives data packets sent by node 2. The node 5 as the source receives only the ACK sent by the node 4. The remaining nodes receive the previous hop and ACK sent from the next hop. As shown in Figure 12, Node 1 and Node 5 have a similar number of packets and acknowledgments. For intermediate nodes (nodes 4, -3 and -2), the number of packets received depends on the number of packets transmitted to them, the packets processed at that node, and the packets forwarded. In this case, in Figure 12, node 4 receives the largest number of packets, then nodes 3 and -2.

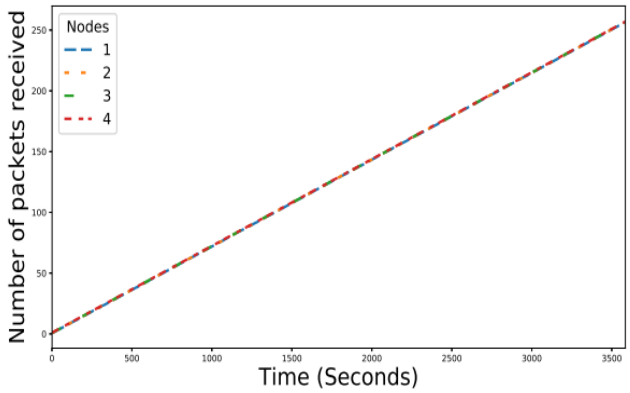

Fig 13: Total packets received without ACK

As shown in FIG. 13, the source node 5 has not received any data packet. Intermediate nodes (nodes 4, -3 and -2) and node 1 receive an equal number of packets sent from their respective previous hops. Therefore, in FIG. 13, it can be observed that the number of data packets received at these nodes overlap.

Depletion of node energy: The multi-line diagrams in Figures 14 and 15 illustrate the timeout energy consumption of all active nodes. Similar to the graph shown in FIG. 8, FIG. 14 here shows a graph of the energy consumption of nodes with ACK hop-to-hop. One well Node-1 has the lowest energy consumption. Since node 1 does not participate in the next hop data packet transmission, it only receives the data packet transmitted by node 2 and returns the ACK to node 2. Nodes 4 and -3 have higher consumption because these nodes have the highest number of send and receive events. In addition, the source Node-5 has higher consumption than Node-2 in sending and forwarding data packets and receiving ACK.

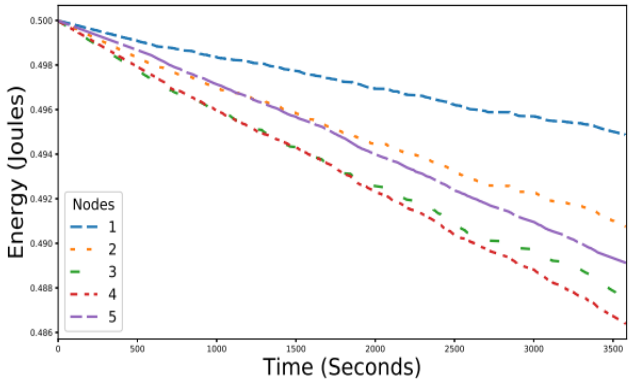

Fig 14: Energy depletion for all active nodes with ACK

The graphs in FIGS. 10 and 12 can be used to check the power consumption shown in FIG. 14 to understand the power consumption during transmission, transmission, retransmission, ACK transmission, and reception of ACK packets. Figure 15 shows the energy consumption curve of the node without ACK. Here, since node 1 is the receiver, it only receives the data packet sent by node 2. Therefore, it has the lowest energy consumption. Intermediate nodes (nodes 4, -3 and -2) overlap because the energy consumption through these nodes is the same.

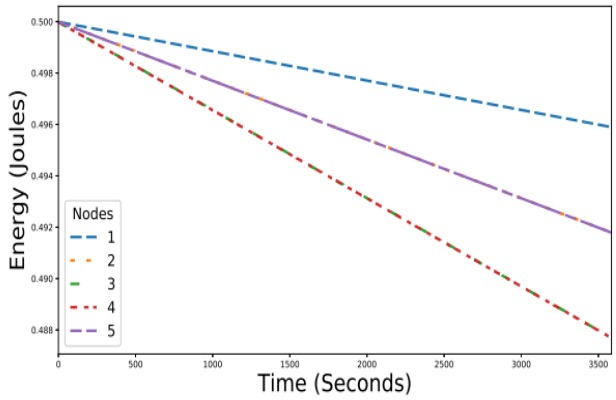

Fig 15: Energy depletion for all active nodes without ACK

Since these nodes are involved in sending and receiving data packets, all data packets are also transmitted without losing data packets due to collisions. Node 5 as the source only participates in sending data packets to its next hop node 4.Since the energy consumed by sending is always greater than the energy consumed by receiving, its depletion rate is higher than node 1 and lower than other intermediate nodes. The graphs in FIGS. 11 and 13 can be used to check the power loss shown in FIG. 15 to see the power loss during the transmission, transmission, and reception of data packets.

VI. CONCLUSION AND FUTURE WORK

UnetStack is one of the most commonly used submarine network simulation tools. The unavailability of its residual energy model has prompted people to design and implement it. The implemented module extends UnetStack's existing HalfDuplexModem implementation for personalized calculation of remaining energy. Using basic and comprehensive simulations, a residual energy model for the number of packets sent and received and the energy consumption of all nodes is demonstrated. Taking into account other parameters of the submarine network (this is the future scope of this work), the energy calculation can still be performed very accurately.

References

[1] G. Han, L. Liu, N. Bao, J. Jiang, W. Zhang, J. J. Rodrigues, and AREP: Reverse routing protocol based on asymmetric links for underwater acoustic sensor networks, Journal of Network and Computer Applications 92 (2017).

[2] K. M. Awan, P. A. Shah, K. Iqbal, S. Gillani, W. Ahmad, Y. Nam, Underwater wireless sensor networks: Recalling recent problems and challenges, "Wireless Communications and Mobile Computing 2019".

[3] M. Jouhari, K. Ibrahimi, H. Tembine, J. Ben-Othman, Underwater wireless sensor networks: Survey on enabling technologies, positioning protocols and the Internet of underwater objects, IEEE Access 7 (2019).

[4] H. Luo, K. Wu, R. Ruby, F. Hong, Z. Guo, L. M. Ni, Simulation and experimentation platforms for underwater acoustic sensor networks: Advancements and challenges, ACM Computing Surveys (CSUR) 50 (2) (2017) 28.

[5] Sehgal, Analysis & simulation of the deep sea acoustic channel for sensor networks.

[6] A. P. Das, S. M. Thampi, Simulation tools for underwater sensor networks: a survey, Simulation 8 (4).

[7] J. Fall, The ns manual, http://www. Isi. Edu/nsnam/ns/ns-documentation.

[8] M. Chitre, R. Bhatnagar, W.-S. Soh, Unetstack: An agent-based softwarestack and simulator for underwater networks, in: 2014 Oceans-St. John's, IEEE, 2014, pp. 1-10.

[9] M. C. Domingo, R. Prior, Energy analysis of routing protocols for underwater wireless sensor networks, Computer communications 31 (6) (2008)1227-1238.

[10] G. Xing, Y. Chen, L. He, W. Su, R. Hou, W. Li, C. Zhang, X. Chen, Energy consumption in NDN's relay underwater acoustic sensor network, IEEE Access 7 (2019) 42694-42702.

[11] to collect energy saving data on the underwater acoustic sensor network, IEEE Systems Journal 12(4) (2018) 3519-3530.

[12] A. Khan, I. Ahmedy, M. Anisi, N. Javaid, I. Ali, N. Khan, M. Alsaqer, H. Mahmood, Routing to minimize interference and energy gaps without localizing the subsea wireless sensor network, Senseurs 18(1) (2018)165.

[13] P. Xie, Z. Zhou, Z. Peng, H. Yan, T. Hu, J.-H. Cui, Z. Shi, Y. Fei, S. Zhou, and Aqua-sim: For ns-2 based simulators for underwater sensor networks, please see: OCEANS 2009, IEEE, 2009, pp. 1-7.

[14] R. Masiero, S. Azad, F. Favaro, M. Petrani, G. Toso, F. Guerra, P. Casari, M. Zorzi, Desert underwater: A framework based on ns-miracle to design, simulate, simulate and implement submarine network protocol test bench, see: 2012 Oceans-Yeosu, IEEE, 2012, pp. 1-10.