Initialization and Routing Optimization for Ad Hoc Underwater Acoustic Networks

Ethem M. Sözer, Milica Stojanovic & John G. Proakis
Northeastern University, Communications and Digital Signal Processing Center
409 Dana Research Building, Boston, MA, 02115
esozer@cdsp.neu.edu / **http://www.cdsp.neu.edu**

ABSTRACT

In this paper, we discuss the design and testing of an underwater acoustic ad hoc network using OPNET Modeler/Radio. The network is intended for the long-term monitoring of a selected ocean area. The data flow of the network is mainly towards a master node, which is responsible for collecting data generated by sensor nodes. When the network is first deployed, an initialization algorithm is executed and preliminary routes are determined. Since the network nodes are battery powered, an important measure of effectiveness is power efficiency. Therefore, in addition to the initialization algorithm, a genetic algorithm based routing optimization that maximizes the network's lifetime is employed. Simulation results are presented.

I. INTRODUCTION

With the advances in acoustic modem technology that enabled high data rates and thus reliable communications, current research focuses on communication between various remote instruments within a network environment. Underwater acoustic (UWA) networks are generally formed by acoustically-connected ocean-bottom sensors, autonomous underwater vehicles (AUVs) and a surface station, which provides a link to an on-shore control center. While many applications require long-term monitoring of the deployment area, the battery-powered network nodes limit the lifetime of UWA networks. In addition, shallow water acoustic channel characteristics, such as low available bandwidth, a highly varied multipath and large propagation delays, restrict the efficiency of UWA networks. Within such an environment, designing an UWA network that maximizes throughput and reliability while minimizing the power consumption is a very difficult task.

A packet transfer protocol that provides reliable transmission of information through an underwater acoustic network was presented in [1]. In this paper, we investigated a dynamic routing algorithm, which consists of an initialization algorithm and a genetic algorithm based routing optimization. In Section II, we give a brief description of the network model. Section III presents the network initialization algorithm. Routing optimization algorithm is discussed in Section IV. Section V gives the OPNET implementation of the initialization algorithm. Section VI and VII present the results and conclusions, respectively.

II. NETWORK MODEL

The network consists of two types of nodes:

- **Sensor Nodes:** These nodes collect data using their sensors. The collected data is then passed to the master nodes through the network. There may be as many sensor nodes as needed depending on the area to be covered.
- **Master Nodes:** Master nodes are responsible for collecting data from sensor nodes. The collected data is then passed to a gateway node that connects the acoustic network to the user on shore.

The sensor nodes are connected to the master node in a hierarchical manner. The number of hops that is required for a sensor node to communicate with the master node determines the level of the node.

III. INITIALIZATION ALGORITHM

Since the network in consideration is an ad hoc network, an initialization algorithm is needed to establish preliminary connections autonomously. The algorithm is based on polling and as such it guarantees connectivity to all the nodes that are acoustically reachable by at least one of their nearest neighbors. During initialization, the nodes create *neighbor tables*. These tables contain a list of each node's neighbors and a quality measure of their link, which can be the minimum required output power level for reliable communications with the corresponding neighbor. The master node then collects the neighbor tables, and form a routing tree. We assume that ID numbers of all the sensor nodes in the network is entered to the master node before deployment. The initialization steps can be listed as follows:

1. The master node sends a polling packet to the first node on the list. The packet contains the unique wakeup sequence and ID of the node that is polled. The master makes the first attempt to send the packet using the minimum output power level. If it cannot get an answer, the output power is increased in steps up to the maximum output power level.

2. If the polled node receives the polling packet, it replies to the master, and is registered as a first level node by the master. In case the master cannot get an answer even with the maximum output power, it is assumed that this node is not a neighbor. The polling for the first level continues for all the nodes and the master node generates a list of the first level nodes.

3. The master node then sends the list of all nodes to one of the first level nodes. The first level node polls the nodes in the list as in steps 1 and 2, and generates a neighbor list. After completion of polling of the last node in the list, the first level node sends its neighbor list to the master node. Instead of sending a list that contains all the nodes in the network to the first level nodes, the master can eliminate the already registered nodes and send a reduced list. We prefer to poll all the nodes at each step to get a complete picture of the network, which will ensure more accurate and efficient operation of the routing optimization.

4. The master now asks the remaining first level nodes to poll their neighbors. After gathering all the neighbor tables from the first level nodes, the master generates a connectivity tree that represents the packet routes. The master also fills the matrices needed for the routing optimization algorithm. During polling, master tries to get the link qualities in both direction, out of and into a node, since the acoustic channel can be asymmetric.

5. The master notifies the registered nodes about the routing tree it has generated and sends the node list to the second level nodes one-by-one for the polling of the remaining nodes. The procedure continues until all the nodes are polled and a complete connectivity matrix with required power levels is generated.

IV. ROUTING OPTIMIZATION

A genetic algorithm based optimization is used to obtain a routing tree that results in minimum energy consumption. The algorithm needs the connectivity graph of the network and the average energy consumption of each node in the network. The connectivity graph, minimum required power for successful packet transmission (P_m) for each link, and the physical length of links are obtained during initialization by polling all the nodes in the network. The master node calculates another required output power (P_c) for each link using the physical length of the links assuming that the channel is an additive white Gaussian noise. Since the underwater channel is time varying, P_m is not a reliable estimate. Therefore, the estimate of average required output power (P_{out}) for each link is obtained as the mean of P_m and P_c. By multiplying P_{out} and average number of transaction, a cost value based on the average energy consumption is calculated for each link. During simulations, offered load is used as the estimate of the average number of sessions per node. The details of the algorithm can be found in [2].

V. IMPLEMENTATION OF THE INITIALIZATION ALGORITHM

The network layer, which is called Layer 3, is responsible for the initialization process. Since the tasks of the master node and the sensor nodes during the initialization stage are different, we created two different Layer 3 processes. The finite state machine (FSM) representations of these processes are given in Figure 1. The Layer 3 process of the sensor nodes can be viewed as a subset of the master node's Layer 3 process.

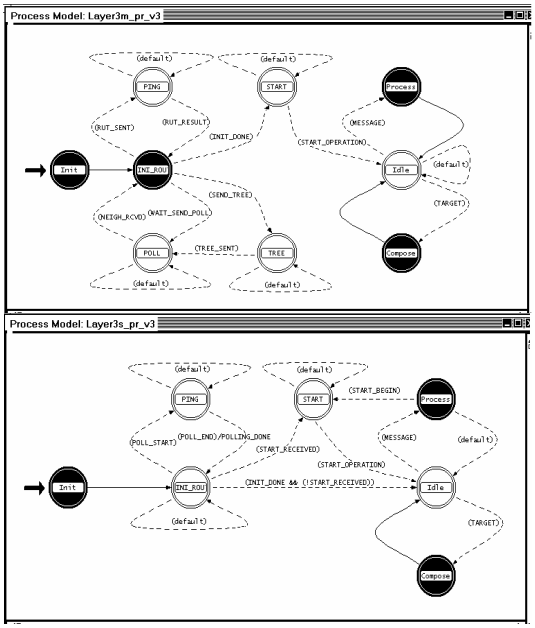

Figure 1. The finite state machine (FSM) representation of Layer 3 processes. The one on the top is the FSM used by the master node, while the one on bottom is used by sensor nodes.

The states of the Layer 3 process and their main functions are listed in the following:

1. INI_ROUTE: This is a forced state that controls the initialization process.
2. PING: This state is used to model the polling of the neighboring nodes. For the master node, neighboring nodes are initially the first level nodes. (After the initialization stage, the dynamic routing algorithm can change the network configuration.)
3. POLL: This state is used to model the polling of the second and higher level nodes.
4. TREE: This state is used to send the routing trees to a newly registered node level after collecting all the neighbor tables from the corresponding level.
5. START: This state is used to send the optimized routing tree and start operation command to all the nodes in the network.
6. COMPOSE: This state is responsible for creating a packet when data becomes available.
7. PROCESS: This state receives data packets from other network nodes and processes them according to the type of the data received.

We also introduced Layer 3 commands that will be passed to Layer 2 together with the Layer 3 packets. During the initialization stage, the packet may have different purposes than carrying data, and Layer 3 needs to notify Layer 2 about the type of the data. In this way, Layer 2 can decide on the transmission scheme. The Layer 3 commands are as follows:

1. DATA
2. PNG (ping)
3. ECH (echo)
4. POLL
5. NEIGHBORS
6. NODE_TREE
7. START

The Layer 2 process implements a data link control protocol that ensures error-free communication between a source and a destination to transport the Layer 3 information sequence [3]. The protocol is based on the MACA protocol [4], which uses RTS-CTS-DATA exchange [1]. However, during the polling stage, we don't need multiple transmissions (like RTS-CTS exchange) to determine if a node is within the transmission range. Such an approach would cause unnecessary energy consumption. Therefore, we established a communication link between Layer 2 and 3 through the Layer 3 commands. For polling purposes, we use two Layer 2 commands:

1. PNG_XMT
2. ECH_XMT

It is assumed that the master node knows the ID numbers of the sensor nodes present in the network. The list of the node IDs is kept in a variable called NodePollList. Also, the neighbor tables of each node are initialized as empty lists called MyNeighborTable. Layer 3 handles the neighbor tables and, when needed, sends a copy of the MyNeighborTable to Layer 2. When the network is activated, Layer 3 of the master node issues a PNG command for the first node in the NodePollList and switches to the PING state. The PNG command and an empty neighbor table are passed to Layer 2. Since the neighbor table is empty, Layer 2 starts the transmissions using the minimum output power level and creates a temporary neighbor table to store the last output power level used. Layer 2 creates a packet that contains the PNG_XMT command and the ID of the node to be polled as the destination ID, and the packet is sent to the acoustic channel. Then Layer 2 switches to the INFO state. If the master node times out, the packet is sent again with 3 dB more output power. This process is continued until the sensor node answers or the maximum output power level is reached.

If the destination node (the node that is being polled) receives the PNG packet, it immediately responds with an ECH packet. The output power level is determined by checking the number of PNG trials that have been sent with the PNG packet. Since the minimum output power level is known to all nodes, the successful output power level can be calculated by using the formula:

$$P_{out}=2^{k}*P_{min}$$

where k is the number of PNG trials. The destination node then returns to its IDLE state. When the master node receives the ECH command, it passes the last used output power level to Layer 3, destroys the temporary neighbor table, and returns to the IDLE state. If the master cannot get a response from the destination, it generates a remote interrupt to notify Layer 3 and returns to the IDLE state. The process is repeated until all nodes in the NodePollList are polled. At this point, Layer 3 switches to the

INI_ROUTE state, where first level nodes are determined, the connectivity graph is filled, average power consumption levels are determined, and neighbor tables are updated.

In addition to determining the minimum power required for successful packet exchange, the range of the sensor nodes are also measured and stored in the neighbor tables. As discussed in Section 4, the instantaneous power levels may not represent the average required power level. A more reliable average can be obtained by using the range information together with the average attenuation in the water as shown in the following formula:

$$P_{est} = (P_{max} * L(r) + P_{last})/2$$

where P_{max} is the maximum output power of the nodes, r is the range between two nodes, and L(r) is the path loss due to propagation through the water at range r [1].

The next phase of the initialization is the POLL state for the master node. At this state, the master node sends the POLL command and the NodePollList to the first level nodes one-by-one, and passes the updated neighbor table to Layer 2. When Layer 2 receives the POLL command, it concludes that the polling of the first level is complete, and fixes its neighbor table until a new routing update. The POLL command is sent using the usual data transmission protocol, which includes an RTS-CTS exchange. Since the first level nodes are not yet initialized, they do not have neighbor tables where the output power levels are registered. To complete the data exchange, the first level nodes need to send back a CTS packet. The output power level for the CTS cannot be determined without a neighbor table. Therefore, the number of trials is sent with the RTS of the POLL command to ensure a minimum number of CTS transmissions.

When an uninitialized sensor node receives an RTS packet, it responds with CTS using the power level determined with the number of trials included in the PNG packet. The destination node creates a temporary neighbor table during the data transmission. The Layer 3 packet that contains the POLL command and the NodePollList is passed to Layer 3 of the sensor node. Since Layer 2 protocol requires acknowledgment and the first sent ACK packet may not be received by the source node, the polling is started with the source node of the POLL command. This also reduces the number of PNG trials, since the destination node already knows the required power level. The source node of the POLL command can also use the PNG packet as an acknowledgment and return to the IDLE state.

Upon receiving the POLL command, Layer 3 of the destination node switches to the PING state, which handles the polling of the nodes as in the case of the master node. The sensor node creates a neighbor table, which contains the IDs and required power levels of its neighbors. The neighbor table is sent to the master node using the NEIGHBOR command.

The correct transmission of the POLL and NEIGHBOR commands is imperative for the proper operation of the initialization algorithm. The reliability of these commands is ensured by setting the maximum number of RTS and DATA transmissions to infinity. During the simulations we saw that for some nodes, seven retries were needed for the correct transmission of these commands, as opposed to a maximum of five retransmissions used for DATA transmission. Another possible approach is to discard the node that did not get the POLL command, or did not respond with a NEIGHBOR command from the level list. The node is set as not registered and the routing tree is updated accordingly.

The master node gathers all the neighbor tables from the first level nodes. The neighbor tables are used to fill the GA matrices and to create the initial routes. The master node then switches to the TREE state and waits for the network to settle before sending any packets. The resulting routing trees are sent to the first level nodes using the NODE_TREE command. The first level node that receives its node tree finishes its initialization and switches to the data communication mode. However, no sensor data is generated and sent to the master node until the master notifies all the nodes that initialization is complete. The master node determines an average delivery time for each NODE_TREE packet depending on the number of hops, and pauses before sending the next NODE_TREE command. This ensures that NODE_TREE packets will not collide with other packets on the channel.

The initialization process continues until all the nodes are polled. When the GA matrices are completed, they are passed to the dynamic

routing algorithm and an optimized routing tree is generated. The final optimized routing tree is sent to the sensor nodes with the START command. This process in carried in the START state. Since the START command carries node tree information, the master node does not send the NODE_TREE command to the last nodes that are registered, but only sends it the START command. Each START command also carries a global start time for the sensor nodes to start data transmission. This global start time is calculated by the master node, as in the case of the NODE_TREE command. The master node sends unique START commands only to the first level nodes. Then, the first level nodes send START commands to their immediate children. This way, the master node does not need to send unique START commands to each node in the network, but uses a flooding type approach instead. The network starts data transmissions only after the global start time is reached. This may happen before all the nodes receive a START command. However, it is enough to prevent excessive packet collisions.

Interested readers can contact the author for the models used in the simulations.

VI. SIMULATION RESULTS

The OPNET simulation program is first run without employing routing optimization. Instead, the network is initialized using the initialization algorithm discussed in Section 5, and the routing tree that is obtained during initialization is used for data transmissions. Figure 2 shows the routing tree for the network for this test case.

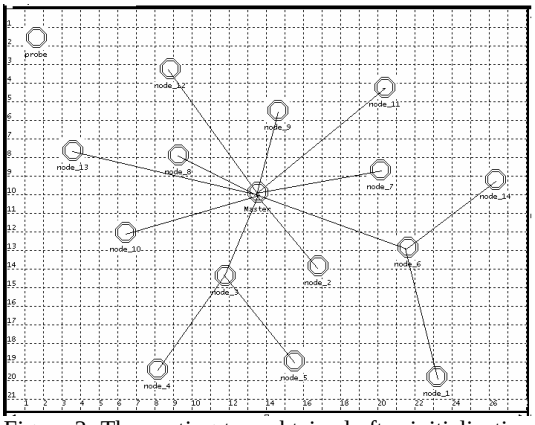

Figure 2. The routing tree obtained after initialization without optimization.

This routing tree is tested for two different offered load values, 0.1e-3 and 1e-3 packets/sec/node, which correspond to approximately 120 and 1200 packets per day, respectively. More than 200 packets per day is unlikely for the network in consideration. The battery levels as a function of time for some typical sensor nodes are shown in Figure 3. The first node fails at 1906414 sec (22 days, 1 hrs, 33 min, 35 sec) for offered load 0.1pk/sec/node, and 309023 sec (3 days, 13 hrs, 50 min, 23 sec) for 1pk/sec/node.

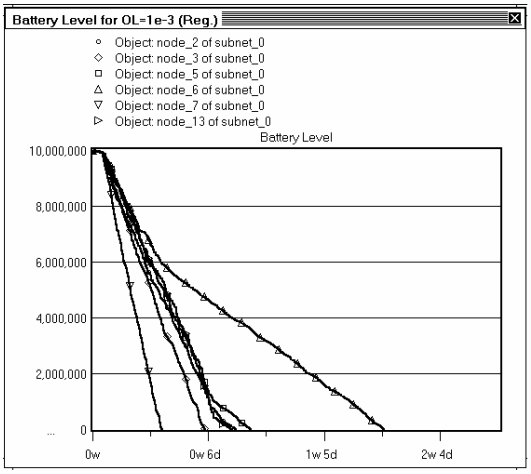

Figure 3. Battery levels of some typical sensor nodes without GA routing optimization for offered load 0.1e-3 packets/sec/node.

Then, the routing optimization is turned on and the simulation is repeated. The optimized trees are given in Figure 4 and 5. The battery efficiency of this configuration is also tested. Figure 6 and 7 shows the battery consumption curves obtained with optimized routing tree. In this case, the first node fails at 3355426 sec (38 days, 20 hrs, 03 min, 46 sec) and 272312 sec (3 days, 03 hrs, 38 min, 32 sec), respectively.

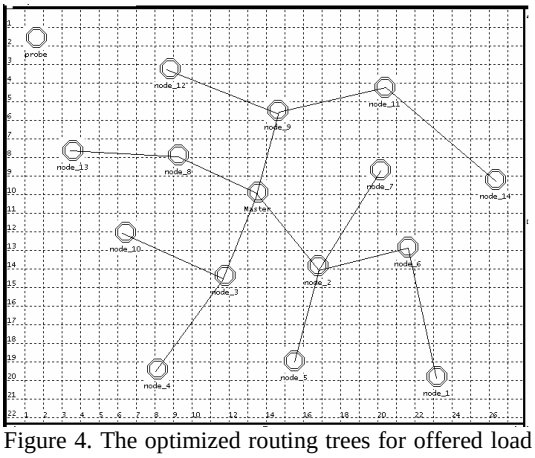

Figure 4. The optimized routing trees for offered load values of 0.1e-3 pk/sec/node.

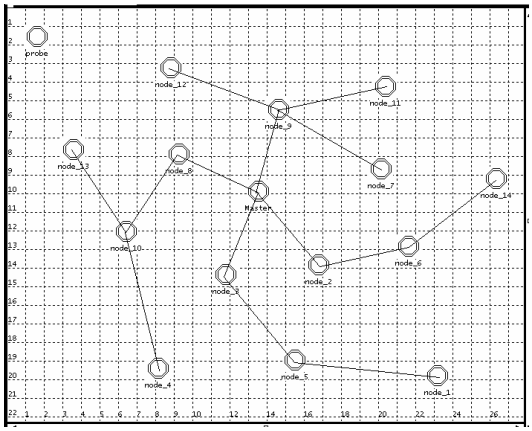

Figure 5. The optimized routing trees for offered load values of 1e-3 pk/sec/node.

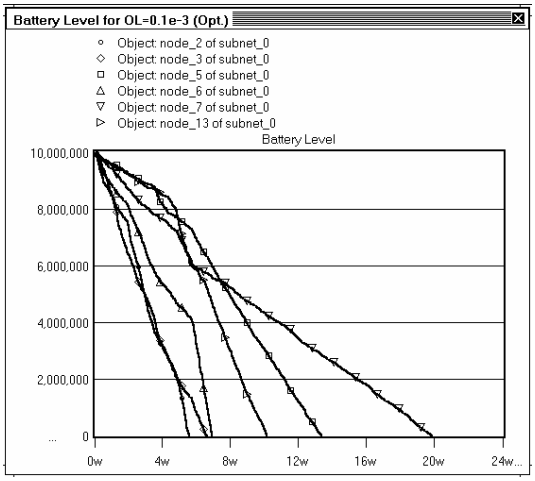

Figure 6. Battery consumption curves of some typical sensor nodes with GA routing optimization for offered load value 0.1e-3 pk/sec/node.

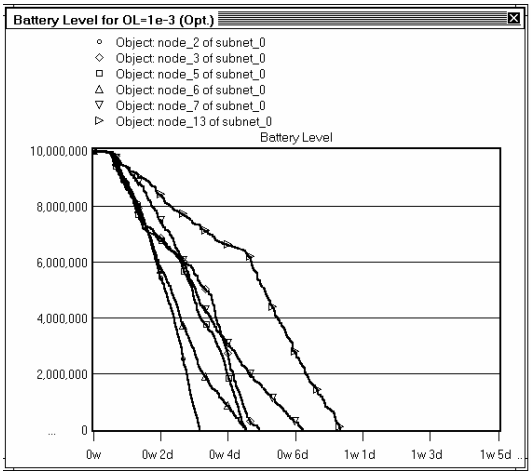

Figure 7. Battery consumption curves of some typical sensor nodes with GA routing optimization for offered load value 1e-3 pk/sec/node.

The failure time of the first nodes as a function of offered load is given in Figure 8. The curve for the optimized case remains above the unoptimized case up to 0.5e-3 pk/sec/node. After this point, the network becomes congested and the assumptions of the expected value of power consumption fail. For lower offered load values, the optimization algorithm extends the lifetime of the network considerably.

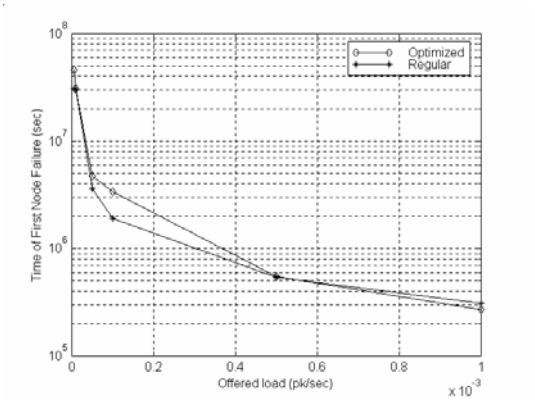

Figure 8. Failure time of the first node as a function of offered load.

We also obtained the throughput as a function of offered load for both cases. The throughput curves are given in Figure 9. As in the case of node failure times, with increasing offered load, the performance of the optimization algorithm degrades. We can also observe that after 0.05e-3pk/sec/node, the network becomes congested and throughput decreases due to excessive packet loss.

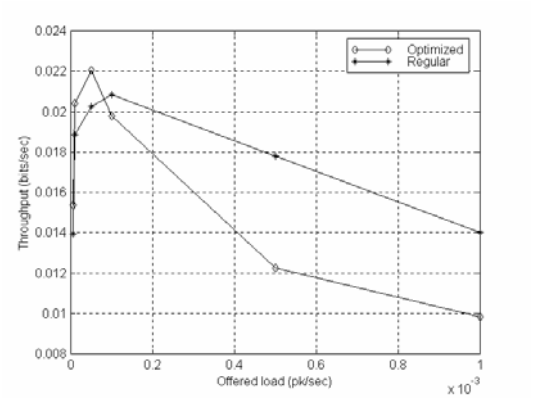

Figure 9. Throughput as a function of offered load for both optimized and unoptimized cases.

Figure 10 shows the end-to-end packet delay as a function of offered load. For offered load values

smaller than 0.6e-3 pk/sec/node, the optimized case results in longer end-to-end delay with respect to the unoptimized case. This is because of the fact that the optimization algorithm favors multiple hops for minimizing energy consumption [5], while for the unoptimized case most of the sensor nodes are directly connected to the master node.

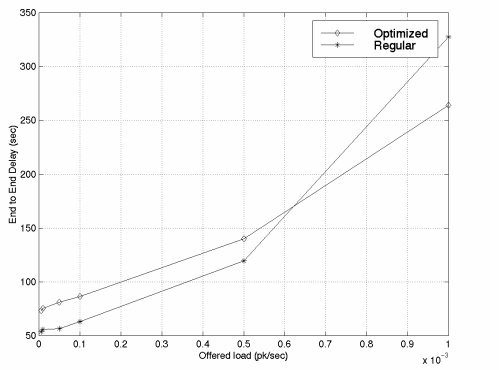

Figure 10. End-to-end packet delay as a function of offered load for both optimized and unoptimized cases.

VII. CONCLUSIONS

An initialization and routing optimization algorithm for an ad hoc underwater acoustic network was tested using OPNET Modeler/Radio. The simulation results show that the initialization algorithm can produce enough information for the routing optimization algorithm. For offered load values that do not result in network congestion, the routing optimization effectively increases the lifetime of the network and throughput. As the network becomes congested, the estimates used by the optimization algorithm fail and the performance degrades. We also showed that to minimize the battery consumption of the network, we need to use multi-hop routes, which results in increased end-to-end packet delay.

VIII. REFERENCES

[1] E. M. Sözer, M. Stojanovic, and John G. Proakis, "Design and simulation of an underwater acoustic local area network," Proc. Opnetwork'99, Washington, D.C., August, 1999.

[2] "Underwater acoustic networks and channel optimization: phase II progress report 6," Technical Report, Delphi Communication Systems, October 1, 1999.

[3] D.Bertsekas and R.Gallager, *Data Networks*, N.J: Prentice Hall, 1992.

[4] P. Karn, "MACA – A new channel access method for packet radio," ARRL/CRRL Amateur Radio 9th Computer Network Conf., Sep. 1990.

[5] E.M. Sözer, M. Stojanovic, and J.G. Proakis, "Underwater acoustic networks," IEEE J. Oceanic Eng., vol. 25, pp. 72-83, Jan. 2000.