I. Introduction

NS2 is an open source network simulator that has become very well used in researching terrestrial wireless network protocols. Underwater Acoustic Networks is a resaerch area of increasing activity. The adaption of the NS2 simulator to underwater acoustic networking may provide a low cost standard by which new underwater protocols can be compared. Several modules for underwater comms have been created with accurate portrayal of the underwater channel as the main goal. These modules work to allow a stable accurate platform for MAC testing and devlopment.

The UAN module is composed of new channel, propagation, phy, and MAC modules. There are two new classes of channel modules, which correspond to the two new styles of propagation model. There are also two new Phy layers, a standard underwater PHY and a PHY designed to approximate the WHOI Micromodem's [citation] implementation of FH-FSK. We have also developed a simple random backoff MAC layer which has been researched with the UAN module and compared to a pure Aloha MAC which has also been included in the UAN module. Each set of modules are described in more detail in the following sections.

Detailed documentation on the code is available at [**??**]. The UAN modules are designed to be minimally invasive to the standard NS2 code. Currently, the only modifications to the NS2 code base are in the mobile node TCL file and the ns-defaults.tcl. A new UanNode class is planned as future work, which will remove the need for both modifications, however it is likely that future MAC work will require the addition of new packets which will require some standard modifications.

The UAN modules distribute responsibility differently than standard NS2 wireless classes. Most notably, collision handling is now handled in the Phy layer as opposed to the MAC layer. This allows for simpler MAC research and development and we feel that it more closely models reality. Information on Packet delivery is still reported the MAC layer in terms of Phy state changes. Also, the Propagation layer is now attached to the channel, and the channel performs initial SNR calculations. The Phy layer, however, still can access the Propagation layer through the channel in order to perform the necessary SINR calculations to fulfill its duty of collision management.

II. Channel Layers

Two channel modules have been developed for UAN simulation. The first is a basic multipurpose channel that interfaces with the Phy and Propagation modules to deliver packets with calculated SNR to other nodes in the network. The second class is similar but is specialized to improve runtime performance while using the Uan/Propagation/BHP that runs Bellhop to get a channel impulse response. The BHP propagation model is described in detail in Section III-B.

A. Channel/Uan

The UAN Channel keeps a list of all nodes in the network using an STL list. When a packet is passed down from a Phy layer, the channel calculates the propagation delay to all other nodes in the network and uses a link to the used propagation model to calculate the SNR at each of the receivers. The channel then schedules an NS2 event to deliver the packet to the receiver Phy layer at the appropriate propagation time. Each packet is delivered to every node in the network regardless of distance. This is not considered to be a problem due to the relative sparseness of most UANs, but if this adversely affects runtime performance, it would be trivial to set a SNR threshold or maximum transmission range to limit the number of affected nodes. This channel can be used by simply setting to "Channel/Uan"

B. Channel/Uan/Bhp

The BHP channel is subclassed from the above channel. It is slightly modified to work with the information returned from Bellhop. Specifically, Bellhop will calculate arrival information for multiple receivers at different ranges and depths in a single execution. Using this feature, the BHP Channel calls a method in the class UanPropagationBhp that will run Bellhop (if necessary) to find the path loss to all receivers from a transmitting node.

III. Propagation Layers

Propagation of acoustic waves underwater is very different from the propagation of RF in air. The difference in propagation characteristics is the largest contributor to a need for a separate simulation platform. To this end we have created (or are planning to create) three basic propagation layers.

All of the UAN propagation models developed so far use the same noise calculations. The calculuations are are based on [**??**]. The required paramters, wind, shipping, and frequency, are bound to TCL variables and may be set as Propagation/Uan/wind, and Propagation/Uan/ship. The frequency value is available from the phy layer.

A. UanPropagationThorp

This layer has not been implemented yet, but will use Thorp's approximation to provide an estimate of pathloss through water. Thorp's approximation has received use in many underwater network simulations to estimate the performance of protocols. The approximation has been shown to be poor at frequencies commonly used in UANs, but it is still an attractive method due to its computational simplicity.

It's implementation is trivial.

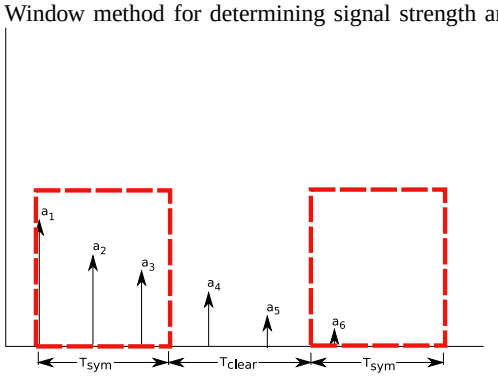
B. UanPropagationBhp

UanPropagationBhp uses the Bellhop ray tracing code available at [**??**]. Bellhop calculates the channel impulse response given a set of environmental parameters. This layer is designed to work with UanChannelBhp, however its use is not required (Performance will be dramatically worse when this layer is used with UanChannel).

A detailed description of how Bellhop works is far beyond the scope of this document. We provide only a quick description of how Bellhop is used to get a channel impulse response, and then how we use that impulse response to calculate the pathloss. Bellhop takes, as an input, a set of envrionmental paramaters. These paramaters include the Sound speed profile, propagation frequency, and surface and bottom characteristics. The environment file also includes the transmitter depth, a set of receiver depths and a set of receiver ranges. Bellhop then creates an arrival file that includes ray arrivals, amplitudes, phase shifts, and delays to all of the receiver range and depth pairs. The arrivals for a receiver depth/range pair can be summed to find the pathloss from the transmitter to the receiver. In all of the simulations conducted thus far we have used the thermocline soundspeed profile described in [**??**].

UanPropagationBhp writes a Bellhop environment file with the thermocline soundspeed profile and a list of receiver depths and ranges (provided by the UanChannelBhp). UanPropagationBhp takes two paramters, the symbol time, $tsym$, and the clear time, $tclear$. When the channel or a Phy layer calls UanPropagationBhp to find the received SNR at a node, UanPropagationBhp sums all arivals that arrive in the interval tsym beginning with the first arrival as shown in figure 1. Any arrivals that arrive between $tsym + tclear$ and $2tsym + tclear$ are considered interference and added to the caluclated noise. Noise calculations are performed in the same way as above. $tsym$ and $tclear$ can be set via TCL using Propagation/Uan/Bhp/tsym and Propagation/Uan/Bhp/tclear respectively.

Fig. 1. Window method for determining signal strength and ISI



The transmission loss, TL, is calculated

$$TL = \sum_{\substack{i \\ \tau_{prop} \le \tau_{di} \le \tau_{prop}+t_{sym}}} |a_i| e^{2\pi f \tau_{di}}$$

The received power is then $P_r = P_{tx} - PL$ where P_{tx} is the transmit power in dB (P_{tx} is set in the simulator by editting the variable Phy/Uan/txpower). And the received SNR (including ISI) is calculated

$$SNR = \frac{P_r}{N + P_{isi}}$$

For τ_{prop}, the propagation delay from the source to the receiver, a_i, the amplitude of the ith multipath arrival as found by Bellhop and τ_{di}, the delay of the ith arrival.

UanPropagationBhp will check the set of arrivals in memory when it is called from UanChannelBhp. If there has been no movement of the transmitting node or the receiving nodes to within predefined tolerances, the previous data will be used. This is a valid approach as Bellhop is a deterministic algorithm. Separate tolerances for frequency, receiver depth, transmitter depth, and range are available. Their corresponding TCL names are available at [**??**]

C. UanPropagationBhf

This section is left for future work. Building on the deterministic model of Bellhop, we will create a cached list of arrivals at varying ranges and frequencies and use these to calculate the arrivals at a specific point. This will create a huge increase in performance. Calculation of SNR will be identical to UanPropagationBhp.

IV. Phy Layers

We have created two Phy layers for the UAN NS2 simulater, UanPhy and UanPhyBhpFsk. The former is a generic Phy layer, and the latter is a derivation that has been designed to closely mimic the FH-FSK implementation found in the WHOI Micromodem [**??**].

The UAN classes depart slightly from standard NS2 Wireless code. In the UAN modules, the Phy layer is responsible for deciding when a packet is decodable and when collisions have taken place. Communication with the MAC layer is handled through state transitions. Currently, these states include TX, RX, and IDLE. Additionally, for carrier sensing applications, dummy transitions are made to CSBUSY and CSIDLE. The Phy layer does not remain in these states, however, when the MAC receives a transition to CSBUSY it knows that energy sensed on the channel is above the CS Threshold and a transition to CSIDLE means that the energy on the channel is no longer above the CS threshold. The Phy layer will transition to one of these states whenever the aggregate SNR on the channel rises above the threshold set in the TCL variable Phy/Uan/csthresh.

A. *UanPhy*

The UanPhy class is a simple Phy layer designed to be a generic platform for MAC layer testing.

In the UanPhy class collision decisions are function of SINR and overlap time. Both of these parameters are accessible from TCL scripts as Phy/Uan/coltime and Phy/Uan/rxthresh. The coltime parameter specifies how long a packet must be below the rxthresh before it is considered lost, and the rxthresh specifies the necessary SINR in dB for acquisition and successful reception.

The SINR is calculated as

$$SINR = 20log(\frac{P_r}{\sum_i P_i + N})$$

Where P_r is the received power, P_i is the power of an interfering packet at the receiver and N is the noise power. Calculation of SINR uses links to the propagation layer to find the noise and receive power.

B. *UanPhyBhpFsk*

The UanPhyBhpFsk layer is designed to work with the UanPropagationBhp (Described in section III-B). layer to mimic the FH-FSK modulation scheme employed in the WHOI Micromodem [**??**].

V. MAC Layers

VI. Results