8/28/17

Aqua-Sim Next Generation

1. Overview:

Aqua-Sim Next Generation (Aqua-Sim NG)is a underwater network simulator which supports a vast amount of protocols and features. Originally developed on the basis of NS-2, Aqua-Sim NG can effectively simulate acoustic signal attenuation and packet collisions in underwater sensor networks (UWSN). Moreover, Aqua-Sim NG supports three-dimensional deployment. This work consists of rewriting and porting Aqua-Sim to NS-3 simulator to improve upon current simulation issues such as memory leakage and usage simplicity. Additionally, this work expands previous underwater support to ensure a more realistic and flexible simulation tool. Work supported by the UWSN Lab at University of Connecticut.

2. Installation:

1. Begin by downloading ns-3 (version 3.26 and 3.24 have been tested with aqua-sim-ng) from www.nsnam.org (https://www.nsnam.org/ns-3-26/download/).

2. Change directory to src within ns-3. Clone aqua-sim-ng from the github repository (https://github.com/rmartin5/aqua-sim-ng).

   ```
   $ cd ns-allinone-3.26/ns-3.26/src/
   $ git clone http://github.com/rmartin5/aqua-sim-ng
   ```

3. Configure ensuring that examples are enabled (while example is using waf, any other ns3 supported configure/build methods will suffice):

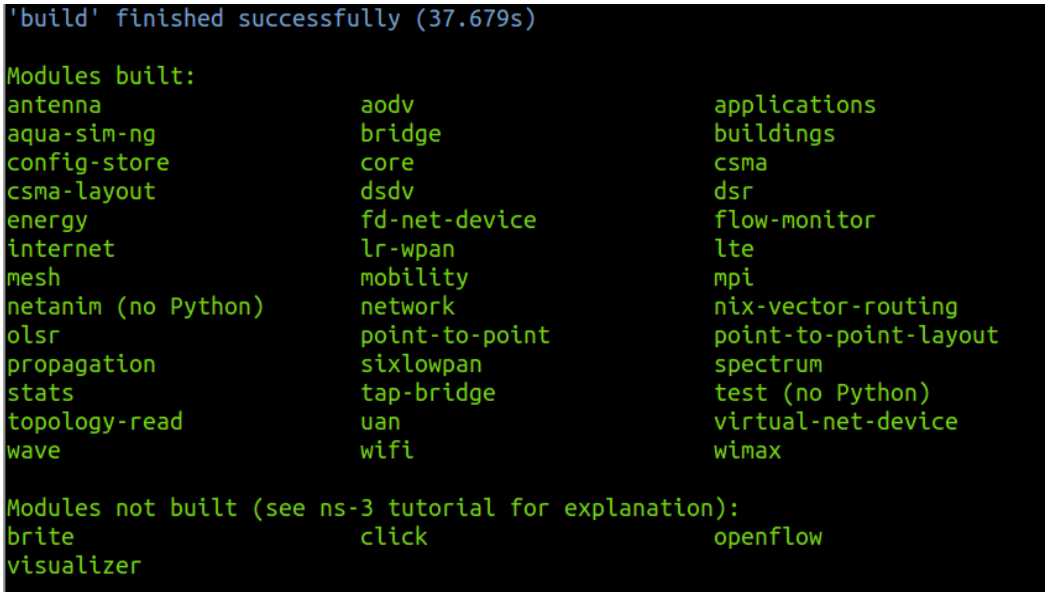
   ```
   $ ./waf –enable-examples --enable-tests configure –disable-python
   ```

   Note: NDN module is using C++11. If using ns-3.24 either remove ndn module through commenting out lines in …/aqua-sim-ng/wscript or including CXXFLAGS="-std=c++0x" during configure such as:

   ```
   $ CXXFLAGS="-std=c++0x" ./waf -d debug --enable-examples --enable-tests configure –disable-python
   ```

4. Build using waf. Once complete you should see a successful message with 'aqua-sim-ng' module within the built section (this will take some time due to the size of NS-3:

   ```
   $ ./waf
   ```



3. Running an Example Script:

Aqua-Sim NG has a diverse range of example scripts which can be used to test different MAC and routing protocols. Alongside this, Aqua-Sim NG overs the capability to test different settings and protocols on lower level components such as PHY and channel. The simplest way to run one of these example scripts can be done through running the following command:

```
$ ./waf --run broadcastMAC_example
```

This command will run the broadcastMAC_example.cc script located in *aqua-sim-ng/examples/* directory. It is important to note that this command should be run from the main directory of ns-3:

```
$ cd ../ns-allinone-3.26.ns-3.26/
```

This is to make Aqua-Sim NG easier to run on NS-3 without having to change scripting files in ns-3's main directory or declare the location of aqua-sim-ng directory.

If you encounter an error at this point, or earlier, make sure you have all necessary depedencies for NS-3 to work properly, spelling of the command is correctly entered, and you are in the correct directory within your terminal.

Without any further logging set up you should only see basic print outs from running this example script. Since this is a proof of concept, we will not pursue exactly what is happening in this example script in this current documentation. Instead you can read through "Example Script Walk Through" to gather a more in depth view of how this script is interacting with the Aqua-Sim NG simulator.

4. Conceptual Overview:

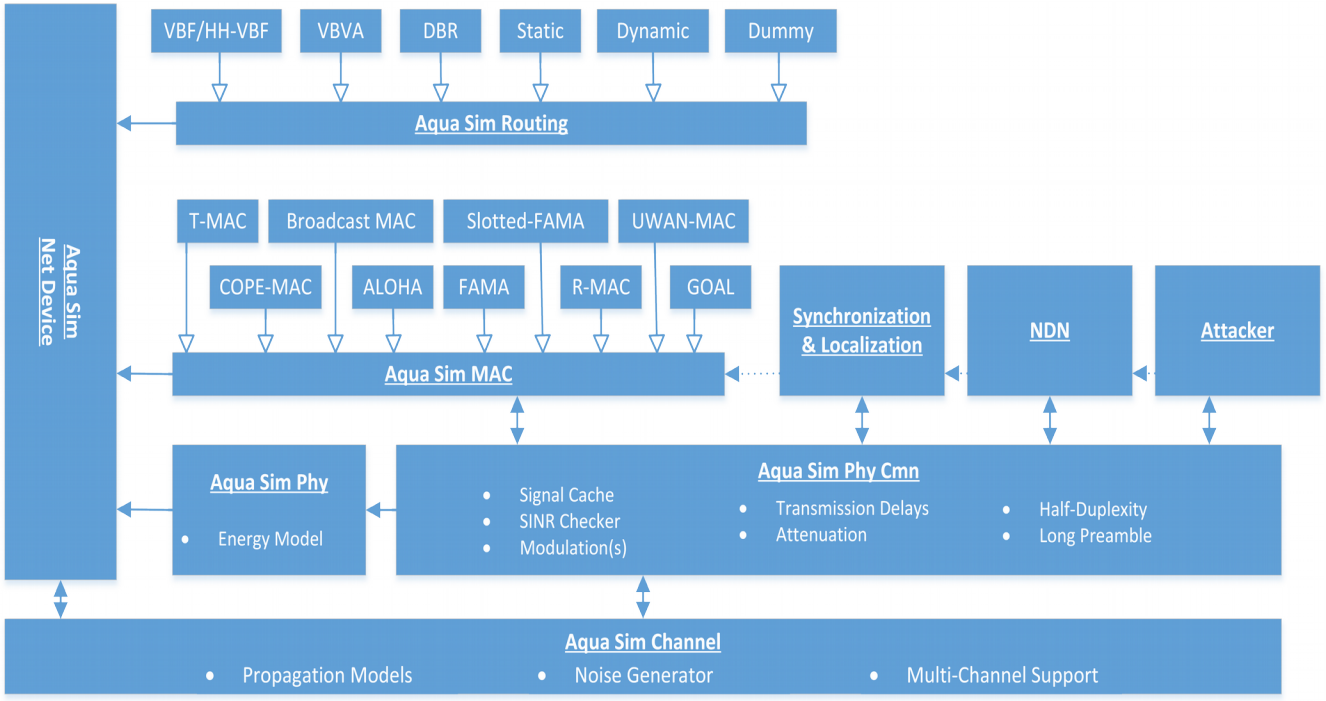


Figure 1: Aqua-Sim NG diagram.

4.1 Headers:

Headers provide another key benefit for ns3, in which only specified headers are used on each packet which eliminates excessively attaching all headers to every packet (and in theory making packets more realistic). Examples of headers used in Aqua-Sim NG can be found in *aqua-sim-header.h*. To handle these headers you can use the preset packet API which allows you to view and change headers as follows:

```
AquaSimHeader ash;
packet->PeekHeader(ash);
```

Here we see an example of declaring and peeking at the given header within a packet. By doing this we can access header information through the variable *ash*, but NOT change their values (i.e. *ash.SetDirection(DOWN)*). This is due to the function call *PeekHeader()* will look at the header's contents, but not deserialize them. In order to change the values within the header we must instead remove the header, followed by adding the header back onto the packet once the changes are made. Here is an example of this:

```
AquaSimHeader ash;
packet->RemoveHeader(ash);
ash.SetDirection(DOWN);
packet->AddHeader(ash);
```

Here we are removing the header from the packet and assigning it to the local header variable. Then we

can set different values for the header variable and finally reattach these headers to the packet. Note that different headers on the same packet must be handled separately and in encapsulated order. Since NS-3 packets are expressed as single buffers for both payload and header space, ordering is very important to avoid data corruption during simulation. Here we show an example of multiple headers:

```
AquaSimHeader ash;
Famaheader FamaH;
packet->RemoveHeader(ash);
packet->RemoveHeader(Famah);
... // edit both the ash and Famah fields
packet->AddHeader(Famah);
packet->AddHeader(ash);
```

By adding the FamaHeader first to the packet, we stay consist with the ordering of packet headers within the current protocol layer and when passing to other layers within the protocol stack.

For Aqua-Sim NG, we push a modularity scheme in which each layer uses its own specialized headers, with the exception of *AquaSimHeader*. This class keeps basic packet information such as packet direction among protocol stack, sender address and destination address. If necessary, a user or developer can print out the current packet's information by using the following within the source code:

```
packet->Print(std::cout);
```

This will call each header's *Print()* function which is currently added to the packet, in the specific order the headers are added. The only requirement for this to work is that printing is enabled within the example script by including the following line:

```
Packet::EnablePrinting();
```

4.2 Packet Handling:

Each protocol layer's base class controls packet handling to other layers. Additionally, since each layer is directly connected with *AquaSimNetDevice*, we originally use this net device to declare what other layers exist in the current simulation protocol stack.

In Figure 2 we show a generic packet flow digram for Aqua-Sim NG. Here we are assuming an application is generating and injecting packets into our simulator. One example of this could be the OnOffApplication, which generates packets of a given size based on a user set data rate. At the MAC layer we use a send queue which collects packets if the modem is in a busy state. Once the modem finishes the current busy state and returns to idle, we begin transmitting all packets within the send queue. For instance, if the modem is currently overhearing a packet destined for another modem, it will still be in the receiving state and therefore can not transmit any packets. Instead, the modem will have to queue any packets that are ready to send and wait until the modem finishes overhearing. In the channel component we are using a function called *ReceivedCopies()*. This function returns all receiving nodes, on the given underwater channel, based on the sender and propagation model in use. On the receiver's end, we use a signal cache model to hold all received packets. Within this model we determine if the current node can decode this packet, and if so, passes the packet to upper protocol layers. The physical layer also manages different underwater packet types, such as synchronization and localization packets, different attacker models, and named data networking packets.

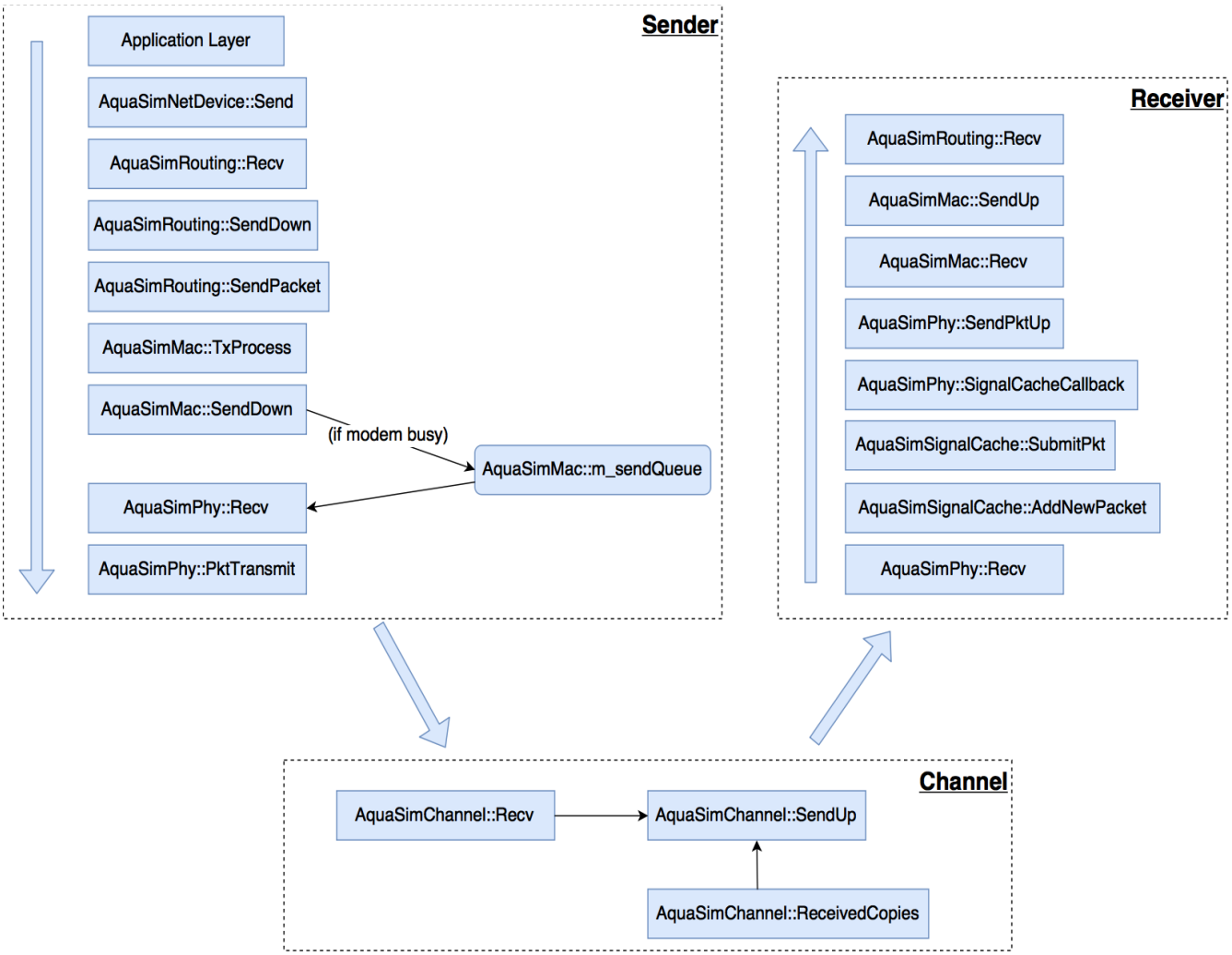


Figure 2: Packet Flow

4.3 Logging and Data Collection:

Being able to see more of what is going on for each module is always an important concept for debugging and depicting event occurrences during runtime. NS-3 Logging provides an in-depth look at the many components to this component of ns3. In short, logging occurs in three steps:

1. Enable logging within a given source file. An example of this from *aqua-sim-phy-cmn.cc* is shown here:

   ```
   NS_LOG_COMPONENT_DEFINE("AquaSimPhyCmn");
   NS_OBJECT_ENSURE_REGISTERED(AquaSimPhyCmn);
   ```

2. Implement logging information within your functions:

   ```
   NS_LOG_FUNCTION(this);
   ```

3. Set logging components, as an additional argument, during example script execution. More details

of this can be found at NS-3's logging tuturial.

```
./waf --run broadcastMAC_example NS_LOG=AquaSimPhyCmn=level_all
```

Or (multiple class logs):

```
export 'NS_LOG=AquaSimPhyCmn=*:NS_LOG=AquaSimChannel=*'
./waf --run broadcastMAC_example
```

Other components, such as saving logging messages to a file, can also be designed for more advanced debugging purposes. Within the example script you can also create a data logger directly to a file. This currently is not set up but can be manually set up (see /ns-3.24/src/examples/tutorial/sixth.cc and seventh.cc for examples of this output file).

There are multiple ways to accomplish data collection in Aqua-Sim NG. The first is using tracers to collect information on a given event. For example, this could be when a sink receives a packet. Both MAC and Routing layers' base class contain tracers which can be implemented to fulfill this user requirement. More information on NS-3's tracing can be found on there website: Tracing. Through the use of logging, you can also collect information which occurs during runtime. A simple example of this is determining how often the energy decrease function is called. Lastly, we have implemented a few basic counters which are currently integrated into the *AquaSimChannel* module. Here we are able to collect data during simulation runtime and directly access the information from our example scripts.

5. Additional Support:

For additional support with running or creating your own example script, please refer to "Example Script Walk Through." Here we will cover some of the components used for creating and populating a underwater network. Additionally, we will cover some of the uses of helper scripts seen in Aqua-Sim NG. For support with protocol creation in Aqua-Sim NG, please refer to "Creating Your Own Protocol in Aqua-Sim NG." Beyond covering some of the basic components of Aqua-Sim NG and NS-3, we will discuss some of the common pitfalls when starting off with Aqua-Sim NG. Furthermore, we will discuss header protocol creation and proper usage, to help avoid unnecessary debugging and incorrect simulation results.

Contact Information:

Questions, comments, issues, or anything else can be directed to:

Robert Martin
Robert.Martin@engr.uconn.edu

or posted directly under aqua-sim-ng issues on GitHub.