12/19/16

Aqua-Sim Next Generation Installation & Start-up

1. Overview:

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

2. Installation:

1. Begin by downloading and installing 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. Pull aqua-sim-ng from the github repository (https://github.com/rmartin5/aqua-sim-ng). Extract file and rename directory 'aqua-sim-ng-master' to 'aqua-sim-ng'.

3. Move aqua-sim-ng directory to the source directory of NS-3:

   …./ns-allinone-3.24/ns-3.24/src/

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

   ./waf --enable-example --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

5. Build. Once complete you should see a successful message with 'aqua-sim-ng' module within the built section:

   ./waf

```
'build' finished successfully (37.679s)

Modules built:
antenna                  aodv                     applications
aqua-sim-ng              bridge                   buildings
config-store             core                     csma
csma-layout              dsdv                     dsr
energy                   fd-net-device            flow-monitor
internet                 lr-wpan                  lte
mesh                     mobility                 mpi
netanim (no Python)      network                  nix-vector-routing
olsr                     point-to-point           point-to-point-layout
propagation              sixlowpan                spectrum
stats                    tap-bridge               test (no Python)
topology-read            uan                      virtual-net-device
wave                     wifi                     wimax

Modules not built (see ns-3 tutorial for explanation):
brite                    click                    openflow
visualizer
```

3. Running Examples:

Currently Aqua-Sim NG has one example within the /aqua-sim-ng/examples directory named *broadcastMAC_example.cc*. To run this example we first assume you have build/configure your ns3 successfully (see NS3 Tutorial<https://www.nsnam.org/docs/release/3.24/tutorial/html/getting-started.html#building-ns3>, specifically 'Testing ns-3' and 'Running a Script' section). Then enter the following command from within your /ns-3.24 directory:

```
robert@robert:~/code/ns-allinone-3.24/ns-3.24$ ./waf --run broadcastMAC_example
```

This will run the given example script. Another helpful area when running your script is to adjust your logging output to either debug or better depict the events occurring during runtime. NS3 Logging<https://www.nsnam.org/docs/manual/html/logging.html> provides an in-depth look at the many components to this component of ns3. In short, logging occurs in three steps:
(1) you enable logging within a given file:

```
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 at execution of script (ensure that 'enable-examples' was set during configure) ('Severity and Level Options' of ns3 logging documentation gives details of each preset macro:

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

Or (multiple class logs):

export 'NS_LOG=AquaSimPhyCmn=level_all:NS_LOG=AquaSimChannel=level_all'
./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).

4. Architecture Components:

While this will not be able to cover all important components of ns3, as the tutorial of ns3 would do, it is meant to give a brief overview to better understand ns3, specifically Aqua-Sim NG, in comparison to typical ns2 Aqua-Sim.

4.1 Layout:

The overall layout of ns3 is meant to consist of a base class and children classes inherited from such. In this way, we see Aqua-Sim NG taking structure of creating an empty base class (such as AquaSimPhy <aqua-sim-phy.h>) with almost all functions being virtual. This means that it is up to the inherited child to determine how to handle these functions. While there are a few set functions within the base class that are constructed for all children to possess (such as SendDown(packet)). Current Aqua-Sim NG work is trying to follow this structure.

4.2 Smart Pointers:

One of the nice advances in ns3 is the implementation of smart pointers. Functions that inherit from Object class are capable of using these smart pointers. The main benefit being they are able to unreference themselves once out of scope, removing many unwanted memory leakage bugs. The unreference process is managed by the pointer class through a counter and Unref function (which should not be called by the user/developer). Pointers are denoted as 'Ptr<--pointer_type--> NameOfPointer'. While many protocol ports from Aqua-Sim 2.0 currently assign pointers to zero, this may not be necessary.

4.3 TypeId Commands:

In substitute for ns2's command(argc,argv) each class can instead have a GetTypeId function similar to the following:

```
TypeId
AquaSimPhyCmn::GetTypeId(void)
{
  static TypeId tid = TypeId("ns3::AquaSimPhyCmn")
    .SetParent<AquaSimPhy> ()
    .AddConstructor<AquaSimPhyCmn> ()
    .AddAttribute("CPThresh", "Capture Threshold (db), default is 10.0 set as 10.",
      DoubleValue (10),
      MakeDoubleAccessor(&AquaSimPhyCmn::m_CPThresh),
      MakeDoubleChecker<double> ())
    .AddAttribute("CSThresh", "Carrier sense threshold (W), default is 1.559e-11 set as 0.",
      DoubleValue(0),
      MakeDoubleAccessor(&AquaSimPhyCmn::m_CSThresh),
      MakeDoubleChecker<double>())
    .AddAttribute("RXThresh", "Receive power threshold (W), default is 3.652e-10 set as 0.",
      DoubleValue(0),
      MakeDoubleAccessor(&AquaSimPhyCmn::m_RXThresh),
      MakeDoubleChecker<double>())
    .AddAttribute("PT", "Transmitted signal power (W).",
```

The idea here is to allow users to set attributes of certain variables at run time through the

example script (see CommandLine cmd; cmd.AddValue(....) within broadcastMAC_example.cc). Each of these variables are set up in a very similar way, with the only main difference being the variable type and how the variable should be accessed. For simplicity we allow generic access to variables (compared to designing if a variable can be Set/Get/ or both). Examples of different AddAttribute types such as pointer or integer can be seen in other source files such as in Aqua-Sim NG. Also note that the layout for each attribute is as such: .AddAttribute("NAME", "DESCRIPTION", "INITIAL_VALUE", "VARIABLE_&", "CHECKER").

4.4 Simulator:

The core simulator of ns3 is quite powerful, of which you are given many capabilities for handling events. You can schedule events with/without context, in the current version of Aqua-Sim only without context is used, while with context would be beneficial for handling packet's payload. One example of scheduling an event can be seen here:

```
Simulator::Schedule(txSendDelay, &AquaSimPhyCmn::SetPhyStatus, this, PHY_IDLE);
```

The layout for Schedule is as followings:

Simulator::Schedule(<DELAY_IN_TIME>,<MEM_PTR>, <OBJ>,<PARAMETERS>)

So in this example we are calling function 'SetPhyStatus' after time 'txSendDelay' with parameter 'PHY_IDLE'. More on ns3's Simulator can be found on their doxygen/documentation.

4.5 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 currently used headers in Aqua-Sim NG can be found in aqua-sim-header.h. These headers are not fully efficient and are mainly structured based on the given Aqua-Sim 2.0 port (including redundancy and wasted header space). To handle these headers you can use the preset packet API which allows you to view and change headers.

```
AquaSimHeader asHeader;
p->PeekHeader(asHeader);
```

Here we see an example of declaring and peeking at the given header within packet p. By doing this we can access header information through the pointer asHeader, but NOT change their values (i.e. asHeader.GetDirection()). In order to change the values within the header we must instead use 'RemoveHeader' as such:

```
AquaSimHeader asHeader;
FamaHeader FamaH;
pkt->RemoveHeader(asHeader);
pkt->RemoveHeader(FamaH);
```

Here we are removing the header from packet 'pkt' and assigning it to the local header variable. Note that different headers on the same packet must be handled separately. Then we can set different values for the header variable (i.e. asHeader.SetDirection(AquaSimHeader::DOWN)) and finally reattach these headers to the packet (i.e. pkt->AddHeader(asHeader)).

5. Helpers:

Within the main directory of aqua-sim-ng there is a folder called 'helper'. Here we include helper classes which are meant to streamline the redundant components of setting up the given simulator within each example script. An example from within aqua-sim-helper.cc is function Create as seen below:

```
Ptr<AquaSimNetDevice>
AquaSimHelper::Create(Ptr<Node> node, Ptr<AquaSimNetDevice> device)
{
  Ptr<AquaSimPhy> phy = m_phy.Create<AquaSimPhy>();
  Ptr<AquaSimMac> mac = m_mac.Create<AquaSimMac>();
  Ptr<AquaSimRouting> routing = m_routing.Create<AquaSimRouting>();

  device->SetPhy(phy);
  device->SetMac(mac);
  device->SetRouting(routing);
  device->ConnectLayers();

  NS_ASSERT(m_channel);
  device->SetChannel(m_channel);

  node->AddDevice(device);

  NS_LOG_DEBUG(this << "Create Dump. Phy:" << device->GetPhy() << " Mac:"
               << device->GetMac() << " Routing:" << device->GetRouting()
               << " Channel:" << device->GetChannel() << "\n");

  return device;
}
```

Here we are setting up each layer and connecting them together. Where AquaSimNetDevice acts as a container for all layer to reside within. Note that we are setting the type of each layer to the base class (i.e. AquaSimPhy) while the actual type being used is set within the helper constructor (i.e. `m_phy.SetTypeId("ns3::AquaSimPhyCmn");`). Furthermore, each layer has a Set function which allows the user to set up multiple attributes of this layer in one command:

i.e. SetPhy(phy, "CPThresh", DoubleValue(10), "CSThresh", DoubleValue(0))

NS_ASSERT is similar to ns2 in which it will halt progression of the simulator in the case that false/0 is the given parameter.

6. Example Script:

(Focused on broadcastMAC_example within 'aqua-sim-ng/examples')

```
NodeContainer nodesCon;
NodeContainer sinksCon;
nodesCon.Create(nodes);
sinksCon.Create(sinks);
```

Similar to Aqua-Sim tcl examples, we must first create a container to hold all nodes and create them accordingly (using the NodeContainer Create function, which creates empty nodes). From here we set up the default sockets and layers (PacketSocketHelper and AquaSimHelper, respectively). Currently we are only supporting a single channel/modulation but in the future this can be expanded to

support a more diverse simulation (i.e. higher SINR in certain channels that may even fluctuate). Next we loop through both sink/regular node containers and populate them with all provided information (this of course can vary based on simulation goals but the current layout sets up default values and static location). Each node is separated by 2000m only on the X axis. This can be expanded using such classes like the given ns3's MobilityModel classes or Aqua-Sim NG's mobility patterns. Sink and regular nodes have no distinction in this version of Aqua-Sim-NG.

The OnOffHelper is used as a traffic generator based on the given attributes DataRate and PacketSize. ApplicationContainer installs this traffic generator on all nodes within nodesCon for the given Start/Stop time. Then we create a socket on the sink (last node within the string network) to receive packets.

Clearly this example is ignoring a lot of factors such as node range, extensive Routing/Mac layers (currently using base class), collisions, energy models, etc. But the generic purpose of this example was to just get a working example of the Aqua-Sim NG simulator and to test the flow of the packets throughout each node's protocol stack (ensuring SendDown/SendUp components are functioning properly).

7. Side Notes:

How to proceed and edit/create a new network layer? The first cause would be setting up the new layer either by creating a new example (referring ns3 examples) or editing an existing example. Depending on the complexity of the new example you may find it helpful to add or create new helper classes to assist in adding attributes of the new layer (see current 'AquaSimChannelHelper' for examples of this with noise generator and propagation model). Additionally you can use the current attributes of each layer when populating a newly created layer or by changing an attribute during simulation run.

Additionally, past protocols on Aqua-Sim 2.0/1.0 have already been ported to Aqua-Sim NG. While these protocols are build-able, they have yet to be fully tested for bugs that may have occurred during porting or resided from past Aqua-Sim iterations.

While much still has to be done, Aqua-Sim NG is trying to adhere to ns3 coding style <https://www.nsnam.org/developers/contributing-code/coding-style/>.

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.