

# Aquasim-Third Generation说明文档

## Aquasim-Third Generation说明文档

介绍  
软件架构  
安装教程  
使用说明  
具体实例

## 介绍

- 总体说明

本项目是吉林大学智慧海洋团队开发的第三代水声网络仿真系统，基于ns3网络仿真系统开发。本项目提供了多种水声组网协议，目前各层仅开放单一协议供测试使用，使用者也可基于此测试框架自行开发。

- 目前已开放协议说明

应用层: onoff-nd-application(位于 ns3/src/aqua-sim-tg/model/ndn 目录下)

传输层: AquaSimTransport(位于 ns3/src/aqua-sim-tg/model 目录下)

路由层: StaticRouting(位于 ns3/src/aqua-sim-tg/model 目录下)

MAC层: Aloha(位于 ns3/src/aqua-sim-tg/model 目录下)

- 未来工作计划

我们会在保持五层网络架构不变的基础上重构整体项目，以增加Aquasim的实用性、真实性。同时我们也会在路由层和MAC层开放更多协议，便于使用者进行测试对比。

## 软件架构

本项目采用经典的五层网络架构。



# 安装教程

本项目上传文件中已包含ns3源码无需另行下载ns3，推荐安装环境为 Ubuntu 18.04.5。

## 1. 安装依赖包

```
vim pre.sh
```

```
#!/bin/sh
sudo apt-get install gcc g++ python python3 -y
sudo apt-get install gcc g++ python python3 python3-dev -y
sudo apt-get install python3-setuptools git mercurial -y
sudo apt-get install qt5-default mercurial -y
sudo apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo python-
pygraphviz python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython
ipython3 -y
sudo apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-dev -y
sudo apt-get install autoconf cvs bzip2 unrar -y
sudo apt-get install gdb valgrind -y
sudo apt-get install uncrustify -y
sudo apt-get install doxygen graphviz imagemagick -y
sudo apt-get install texlive texlive-extra-utils texlive-latex-extra texlive-
font-utils texlive-lang-portuguese dvipng latexmk -y
sudo apt-get install python3-sphinx dia -y
sudo apt-get install gsl-bin libgsl-dev libgsl23 libgslcblas0 -y
sudo apt-get install tcpdump -y
sudo apt-get install sqlite sqlite3 libsqlite3-dev -y
sudo apt-get install libxml2 libxml2-dev -y
sudo apt-get install cmake libc6-dev libc6-dev-i386 libclang-dev llvm-dev
automake -y
sudo apt-get install python-pip -y
pip install cxxfilt -y
sudo apt-get install libgtk2.0-0 libgtk2.0-dev -y
sudo apt-get install vtun lxc -y
sudo apt-get install libboost-signals-dev libboost-filesystem-dev -y
sudo apt-get install python-dev python-pygraphviz python-kiwi python-pygoocanvas
python-gnome2 gir1.2-goocanvas-2.0 python-rsvg -y
```

```
chmod +x pre.sh
./pre.sh
```

## 2. 进入ns3目录下配置

```
./waf configure
```

## 3. 在ns3目录下编译运行

```
./waf
./waf --run hello-simulator
```

执行 `./waf` 之后出现如下界面即可，注意要着重检查 `aqua-sim-tg` 模块是否能正确编译。

```
Build commands will be stored in build/compile_commands.json
'build' finished successfully (8.530s)

Modules built:
antenna          aodv          applications
aqua-sim-tg      bridge       buildings
config-store     core         csma
csma-layout      dsdv        dsr
energy          fd-net-device flow-monitor
internet         internet-apps 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 traffic-control
uan             virtual-net-device wave
wifi           wimax

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

执行 `./waf --run hello-simulator` 之后出现如下界面即可。

```
ch@ubuntu:~/ns3$ ./waf --run hello-simulator
Waf: Entering directory `/home/ch/ns3/build'
Waf: Leaving directory `/home/ch/ns3/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (1.186s)
Hello Simulator
```

## 使用说明

### 1. 总体说明

aqua-sim-tg默认使用静态路由协议，路由表为 ns3/1.txt 文件，路由表项目格式为（表项所属节点地址：目的地址：下一跳地址）。例如表项 1：2：3表示节点1收到一个目的地址为2的包其下一跳为节点3。

具体协议文件位于 ns3/src/aqua-sim-tg/model 目录下。

用户运行脚本位于 ns3/scratch 目录下。

### 2. 增加协议

在 ns3/src/aqua-sim-tg/model 目录下增加协议源文件，在 ns3/src/aqua-sim-tg/wscript 文件中写入新增协议文件名。

### 3. 运行脚本

编写脚本之后，在 ns3 目录下使用 `./waf --run fileName` 命令运行

## 具体实例

下面，我们将用一个具体的例子来展示如何编写运行脚本。在下面的例子中，我们组织了一个5节点的链状网络，并让四号节点作为接收节点（line 56），但需要注意的是，这里的四号指在NodeContainer中的下标为4（下标从零开始），即第五个节点。

```
int main(int argc, char *argv[])
{
    double simStop = 300000; //仿真结束时间
    int nodes = 5; //节点个数
```

```

LogComponentEnable("AquaSimAloha", LOG_LEVEL_DEBUG);

CommandLine cmd;
cmd.AddValue("simStop", "Length of simulation", simStop);
cmd.AddValue("nodes", "Amount of regular underwater nodes", nodes);
cmd.Parse(argc, argv);

std::cout << "-----Initializing simulation-----\n";

NodeContainer nodesCon;
nodesCon.Create(nodes);

AquaSimSocketHelper socketHelper;
socketHelper.Install(nodesCon);

AquaSimChannelHelper channel = AquaSimChannelHelper::Default(); //默认信道模型

AquaSimHelper asHelper = AquaSimHelper::Default();
asHelper.SetChannel(channel.Create());
asHelper.SetRouting("ns3::AquaSimStaticRouting"); //设置路由协议
asHelper.SetMac("ns3::AquaSimAloha"); //设置MAC协议

MobilityHelper mobility;
NetDeviceContainer devices;
Ptr<ListPositionAllocator> position = CreateObject<ListPositionAllocator>();

Vector boundry = Vector((1000), (1010), 100);

std::cout << "Creating Nodes\n";
int k = 0;

for (NodeContainer::Iterator i = nodesCon.Begin(); i != nodesCon.End();
    i++)
{
    //生成节点位置, 设置节点net device和地址, 地址从一开始编号
    Ptr<AquaSimNetDevice> newDevice = CreateObject<AquaSimNetDevice>();
    position->Add(boundry);
    devices.Add(asHelper.Create(*i, newDevice));
    boundry.x = 1000;
    boundry.y = boundry.y + 1000;
}
std::cout << "Creating Nodes End\n";

mobility.SetPositionAllocator(position); //设置节点位置
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(nodesCon);

AquaSimSocketAddress socket;
socket.SetAllDevices();
socket.SetDestinationAddress(devices.Get(4)->GetAddress()); //设置接收节点编号

OnOffNdHelper app("ns3::AquaSimSocketFactory", Address(socket));
app.SetAttribute("OnTime",

StringValue("ns3::ExponentialRandomVariable[Mean=100|Bound=0.0]")); //设置发包率服从
泊松分布
app.SetAttribute("OffTime",

```

```
StringValue("ns3::ConstantRandomVariable[Constant=0]"));

ApplicationContainer apps = app.Install(nodesCon);
apps.Start(Seconds(400)); //设置仿真开始时间
apps.Stop(Seconds(simStop)); //设置仿真结束时间

Packet::EnablePrinting();
std::cout << "-----Running Simulation-----\n";
Simulator::Stop(Seconds(simStop));
Simulator::Run();
Simulator::Destroy();

std::cout << "fin.\n";
return 0;
}
```