Mobile Services and Networks

Grupo de Arquitectura y Redes de Computadores

Introduction to simulation in NS3

ISRN ARCO/--2020/

3 de febrero de 2020

1. Simulating mobile communications

1.1. Objectives

The main objective of this documents is to guide student in its first simulation process through NS3 simulator.

1.2. Prerequisites

Meanwhile redacting this document, the most recent stable release of NS3 simulator is the version 3.30.1. The student is free to use any other version of the simulator. We work with basic examples and commands so probably the following procedure is also valid for next releases of ns3.

In a similar way, the steps that we will describe later are tested in a PC with Debian GNU/Linux installed.

1.3. Introduction

The student should follow this procedure

- Download the all-in-one package from www.nsnam.org (Direct link)
- Unzip the file ns-allinone-3.30.1.tar.bz2

  ```
  $tar -xjvf ns-allinone-3.30.1.tar.bz2
  ```

- Install the ns3 following this full list of dependencies , as a summary, in debian:
  - pre-requisites (as root):

    ```
    apt-get install gcc g++ python python3 python3-dev
    apt-get install mercurial python-setuptools git
    apt-get install qt5-default sqlite sqlite3 libsqlite3-dev
    apt-get install gir1.2-goocanvas-2.0 python-gi python-gi-cairo python-pygraphviz
    apt-get install libgtk2.0-0 libgtk2.0-dev
    apt-get install libxml2 libxml2-dev
    apt-get install doxygen graphviz imagemagick
    apt-get install texlive texlive-extra-utils texlive-latex-extra
    apt-get install python-sphinx dia
    apt-get install gdb valgrind
    apt-get install gsl-bin libgsl-dev libgsl23 libgslcblas0
    ```

- go to the directory (ns-allinone-3.30.1) and execute build.py with the following options:

  ```
  ./build.py --enable-test --enable-examples
  ```

- you should get something like *'build' finished successfully (8m45.341s)* with a list of modules built and a list of modules not built. At this point you could start to work with ns3, however, the most interesting source of information for starting are examples and test. We need to enable examples and test for use them.
- Execute the verification script to check that everything is ok. Inside of ns-3.30.1 directory execute test.py. You should see something like:

  ```
  .
  .
  281 of 284 tests passed (281 passed, 3 skipped, 0 failed, 0 crashed, 0 valgrind e
  ```

At this point, you are ready for start to execute simulations from examples.

1.4. Simulation flow

In *ns-3.30.1/examples/stats* you can find a good example of a full simulation process including:

- a wireless topology defined mainly in wifi-example-sim.cc. The topology defines two nodes with a 802.11 interface each one. Both nodes communicate each other by mean a wireless channel sending packets from attached applications (implemented in wifi-example-apps.cc/h). The simulation takes several arguments from line command. The most important argument for us is the distance apart to place nodes in meters. This simulation put all results in a sqlite3 database for be analysed.
- the shell script wifi-example-db.sh executes the simulated topology changing distance between wireless nodes. After the execution, the script open the database with the results, summarises the results in wifi-default.data file and executes a gnuplot script which build a graph.

The graph shows the amount of packet lost according to the distance between the nodes (figure 1.1).

We will use this example extensively (removing, changing and adding source lines of code) so you should do a backup copy in order to recover the original source code whenever you want.

1.5. Lab 1 deliverable

The student should write a report with the results of the steps redacted in each subsection of this section. So, for each subsection, a graph with the result obtained should be embedded in the report. The student also should answer each one of the subsection's questions.

1.5.1. Changing the script

Starting with the original example stats:

- to increment the distances used in each iteration of the simulation to: 25 50 75 100 125 145 147 150 152 155 157 160 162 165 167 170 172 175 177 180 185 190 195 200 210 220 230 240 250 300 350 400 450 500 600 750 1000
- to reduce the number of iterations for each distance from five to one.
- to adjust the x-axis of the gnuplot script to show results of the new distance.

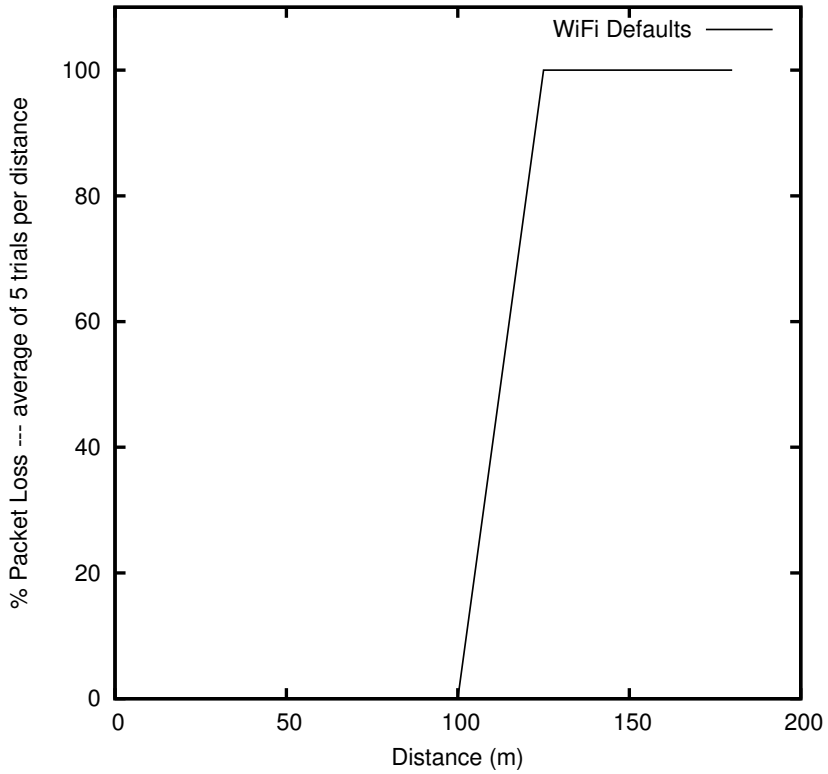

FIGURA 1.1: Figure 1: Graphic generated by example stats of NS3

1.5.2. Changing the antenna parameters of simulation

Let's start changing the topology in the physical layer. The next source code correspond graphically with figure 1.2. In the lines 1 and 2 a node container with two nodes is created (1.2.a). A WifiHelper is a class for connect all elements in a wifi environment. Lines 5-6, 7 and 8 are devoted to create, conceptually, a wifi mac card, a wifi physical equipment (equivalent to an antenna) and a wireless channel respectively (1.2.b)). Finally, first we have to attach the channel to the physical layer 1.2.c in the line 9 and next, install in each node, the combination of wifiPhy and wifiMac (conceptually the wifi interface and the antenna) using for that the wifiHelper (line 10).

```
1  NodeContainer nodes;
2  nodes.Create (2);

3  NS_LOG_INFO ("Installing WiFi and Internet stack.");
4  WifiHelper wifi = WifiHelper::Default ();

5  NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
6  wifiMac.SetType ("ns3::AdhocWifiMac");
7  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
8  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
9   wifiPhy.SetChannel (wifiChannel.Create ());
10  NetDeviceContainer nodeDevices = wifi.Install (wifiPhy, wifiMac, nodes);
```

Our main target in this subsection is to change antenna parameters, so:

- which object represents the antenna? Which properties do you should change for increase the range of a successful communication?

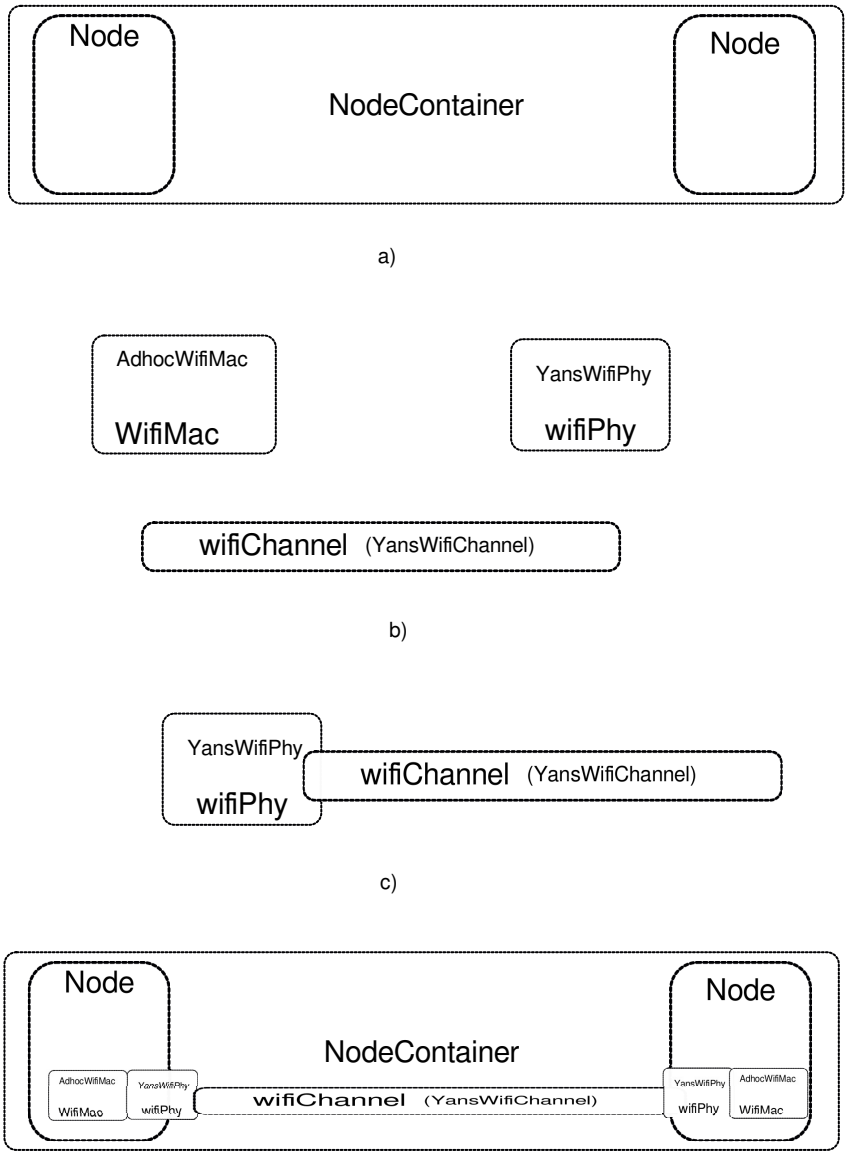

FIGURA 1.2: NS3 objects

- To study the values of the properties to change in order to establish a successfully communication of 200, 400 and 800 meters. The student should include a table with parameters changed to get a successful communication and the proof of the working configuration (include also the corresponding graphics)

1.5.3. The propagation model

As the reader can observe, the lost of connectivity (100 % of packet loss) is achieved in abrupt way. For simulating high level protocols is ok since they need keep simple low-level communications issues. However, This lost of connectivity is not very realistic according to the real wifi communication where the lost of connectivity is done in a soft way. So, it would be more accurate increasing the rate of packet lost according to the distance between both nodes.

In NS3 one of the main issues is modelling the radio wave propagation. Researchers has been developing accurate models for simulate radio propagation models. In [PS12] you can find a brief description of the most used in wireless simulation.

The key problem in a radio propagation model is calculate how distance, fading, etc. affects to the power of the radio wave. In NS3 this is the main issue of the channel class where you have to add a propagation loss according to the model you want to use. Coming back to our first example, you can see how in line 8 a default wifi channel is created (variable wifiChannel) and is attached to the physical interface (line 9) with defaults parameters.

```
1  NodeContainer nodes;
2  nodes.Create (2);

3  NS_LOG_INFO ("Installing WiFi and Internet stack.");
4  WifiHelper wifi = WifiHelper::Default ();

5  NqosWifiMacHelper wifiMac = NqosWifiMacHelper::Default ();
6  wifiMac.SetType ("ns3::AdhocWifiMac");
7  YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
8  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
9   wifiPhy.SetChannel (wifiChannel.Create ());
10  NetDeviceContainer nodeDevices = wifi.Install (wifiPhy, wifiMac, nodes);
```

In this section:

- Surf in the NS3 documentation and list the different types of loss propagation models implemented.
- Test one of the radio propagation models using our example (check AddPropagationLoss method of class YansWifiChannelHelper) and provide a graphic comparing the results of the YansWifiChannel with default loss propagation model and the radio propagation model chosen by you. Feel you free of using several propagation models.

Disable the antenna modification done in the last section in order to have defaults settings for the configurated loss model. In the same way increase the number of test in the area where rate of packet lost stats to increase (for example testing every 10 meters between 100 and 200 meters if you observe that between these two distances the loss of packets start to be different of 0).

Bibliografía

[PS12] Ramesh C. Poonia y Vikram Singh. Performance evaluation of Radio Propagation Model for Vehicular ad-hoc networks using vanet mobisim and ns-2. *International Journal of Distributed and Parallel Systems (IJDPS)*, 2012.