

Technical Note

Creating a new model in NS3 Network Simulator

Lesly Maygua-Marcillo¹, Luis Urquiza-Aguilar¹, Martha Paredes-Paredes¹

¹ Departamento de Electrónica, Telecomunicaciones y Redes de Información, Facultad de Eléctrica y Electrónica, Escuela Politécnica Nacional (EPN), C. Ladrón de Guevara E11-253, Quito PO.Box 17-01-2759, Ecuador; lesly.maygua@epn.edu.ec, cecilia.paredes@epn.edu.ec.

* Correspondence: luis.urquiza@epn.edu.ec; Tel.:+593 2297-6300 ext.2311

Abstract: Nowadays, network simulators are very useful to model a communications system. In this technical note we will focus in the creation of a new error model in NS-3 network simulator. This note describes the main steps to create or modify an Wi-Fi error model in this network simulator. In our case, we have created a new error model to included the approach of [1] to compute PER (Packet Error Rate) for vehicular environments.

Keywords: Error model, New model, NS-3, VANET, 802.11p

1. Introduction

Usually in communications research projects, researchers look for the way to create or modify protocols, methods, process, among others in the chosen simulation tools . Nowadays, Wi-Fi is one of the most used technologies in the wireless communications environments. Currently, the Wi-Fi module of NS-3 includes three error models, which are used in different scenarios. For this reason, in this technical note we show a way to add a new model in the popular NS-3 network simulator. For our case, we show how to add a new error model in the Wi-Fi module [2]. The new error model to be implemented was the proposed in [1]. This approach has the advantage of being a easy computation of PER.

Although in this technical note we focus on an error model, it should be noted that to create any type of model the process is basically the same. Notice that, we will use NS-3 version 25 and the Ubuntu distribution 14.04.5 LTS. In the next sections, we explain the process in detail

2. Adding a New Model

It must be clear to create a new model in the NS-3 network simulator is that "a module is not the same that the model" [3] [4]. The NS-3 models are simply abstract representations of real-world objects such as devices, channels, protocols, etc. On the other hand, an NS-3 module can include more than one model. For example, the Internet module contains models for TCP and UDP [5].

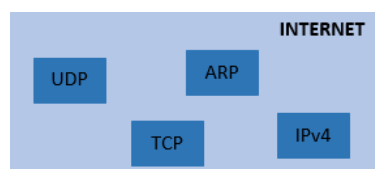


Figure 1. Internet Module, with some with some of its models

In the Fig. 1, we show the Internet module in NS-3 with some of their models. Here we can clearly differentiate the module and model. The processes to create each one is quite different.

The first step in creation process of a new model should be considering the general aspects about the structure of the new model, for example:

- To consider the attributes, configurations.
- To create a modular model.
- To consider dependencies and linked objects.

In our case, as it is an error model, we will analyze the existing objects linked to the other error models, in order to create a similar dependency when adding our new model.

2.1. Configuration

To configure the new model, we recommend the following steps:

- 1 . Go to the module in which you want to create the new model, in our case we will work in Wi-Fi module.
- 2 . Create the header file (.h): In this file is necessary to create a new class for the new model considering the inheritance. And the linked objects. The Fig. 2 show us this new class.

```
namespace ns3 {
/**
 * \ingroup wifi
 *
 * A model for the error rate for different modulations. For OFDM modulation,
 * the model description and validation can be found in
 * http://www.nsnam.org/~pei/80211ofdm.pdf. For DSSS modulations (802.11b),
 * the model uses the DsssErrorRateModel.
 */
class NewErrorRateModel : public ErrorRateModel
{
public:
    static TypeId GetTypeId (void);

    NewErrorRateModel ();
};
```

Figure 2. New class `NewErrorRateModel`, inheritance of `ErrorRateModel`

- 3 . Create the source code .cc: In this file we should code all the processes in functions that we want the model to execute.
- 4 . Enable the new model. In our case, due to the new model created is part of "Wi-Fi Helper", we should modify this Wi-Fi Helper model to enable the new model. Fig. 3 shows the command used to enable the model.

```
LogComponentEnable ("MacRxMiddle", LOG_LEVEL_ALL);
LogComponentEnable ("MsduAggregator", LOG_LEVEL_ALL);
LogComponentEnable ("MsduStandardAggregator", LOG_LEVEL_ALL);
LogComponentEnable ("NistErrorRateModel", LOG_LEVEL_ALL);
LogComponentEnable ("NewErrorRateModel", LOG_LEVEL_ALL);
LogComponentEnable ("OnoeWifiRemoteStation", LOG_LEVEL_ALL);
LogComponentEnable ("PropagationLossModel", LOG_LEVEL_ALL);
LogComponentEnable ("RegularWifiMac", LOG_LEVEL_ALL);
LogComponentEnable ("RraWifiManager", LOG_LEVEL_ALL);
```

Figure 3. Enabling the model, in Wi-Fi Helper

2.2. Storage

We should decide where in the Source Tree where the Model will be located.

All of the NS-3 model source code is in the directory `src/`. You will need to choose which subdirectory your model will resides in [6]. Therefore, it makes sense to put it into the `src/` directory somewhere, particularly for ease of integrating with the build system.

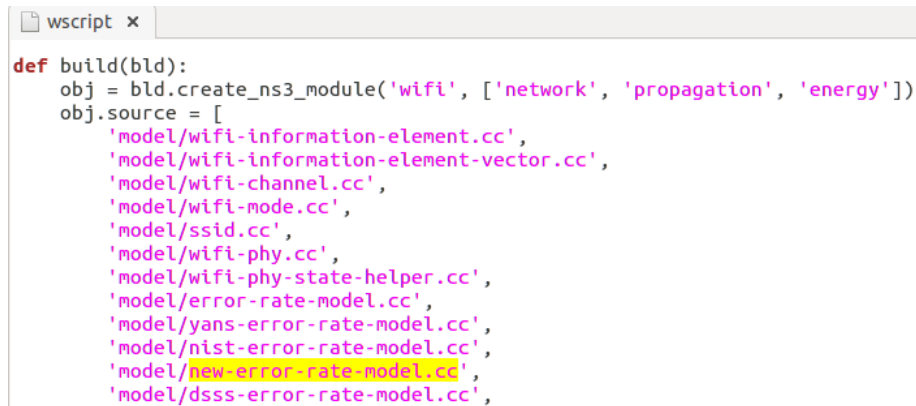
In the case of the error model, so it makes sense to implement this in the `src/Wi-Fi/` module where NS-3 error models are implemented [7].

2.3. Building

NS-3 network simulator uses the "waf" build system. So, you will have to integrate your new NS-3 model using Waf build system. This requires that you add your files to the wscript file located in each directory.

Wscript is very important, because if we do not add the new model here, the simulation will not be executed or if it does then it will be wrong.

Let's start by adding error-model.h and error-model.cc (the could be empty) to wscript in src/Wi-Fi. It is really just a matter of adding the .cc file to the rest of the source files, and the .h file to the list of the header files. The Fig.4 show us how to edit the file.



```

wscript x
def build(bld):
    obj = bld.create_ns3_module('wifi', ['network', 'propagation', 'energy'])
    obj.source = [
        'model/wifi-information-element.cc',
        'model/wifi-information-element-vector.cc',
        'model/wifi-channel.cc',
        'model/wifi-mode.cc',
        'model/ssid.cc',
        'model/wifi-phy.cc',
        'model/wifi-phy-state-helper.cc',
        'model/error-rate-model.cc',
        'model/yans-error-rate-model.cc',
        'model/nist-error-rate-model.cc',
        'model/new-error-rate-model.cc',
        'model/dsss-error-rate-model.cc',
    ]
  
```

Figure 4. Editing WSCRIPT

In addition, in the NS-3.25 directory we could type "./test.py". You should not have broken anything by this operation.

2.4. Using the model

We can call the new error model by adding the following configuration. Fig. 5 show a configuration by default.

```

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper();
wifiPhy.SetErrorRateModel ("ns3::NewErrorRateModel");
  
```

Figure 5. Configuration of NewErrorModel

3. NewErrorModel

The objective of our `NewErrorModel` in NS-3 is to add an Analytical PER (Packet Error Rate) Model for 802.11p networks.

For that, we have modified the main operation of existing models. We have based on (1) proposed on [1]. Eq. 1 shows the PER as function of SNR and packet length.

$$PER_R(SNR, l) = \frac{1 - \tanh(a_R(l) - b_R(l)(SNR + c))}{2} \quad (1)$$

$$a_R(l) = c_1 e^{d_1 \cdot l} + c_2 e^{d_2 \cdot l} \quad (2)$$

$$b_R(l) = c_3 e^{d_3 \cdot l} + c_4 e^{d_4 \cdot l} \quad (3)$$

Where c is a constant to sustain interpolation issues, its value is 10 dB. The values of: c_i , d_i ($i = 1, \dots, 4$) are reported in Table I in [1]. Eq. 1 was added to the source file `NewErrorModel.cc`.

4. Simulation results

For the simulations we use an wireless scenario for MANETs (Mobile Ad-hoc Networks) with different parameters according to Table 1. To test the variation we have compared the three following error models performance: `YansErrorRateModel`, `NistErrorRateModel` and `NewErrorRateModel`.

Table 1. Simulation parameters

Parameter	Scenario 1	Scenario 2
Number of nodes	60	25
Mobility model	ManhattanGrid	RandomWaypoint
Average speed nodes	1.8 km/h	5 km/h
Area	300 x 150 m	500 x 500 m
Propagation model	LogDistance	LogDistance
Power transmission	16.0206 dbm	16.0206 dbm
Packet size	1024 bytes	1024 bytes
Routing protocol	AODV	AODV

Here, we show a comparison between three error models using two scenarios. In the first case we have tested the models under a MANET with 60 nodes using a Manhattan Grid mobility model. The Fig. 6 and Fig. 7, scenarios 1 and 2 respectively, show us the throughput obtained in the transmission.

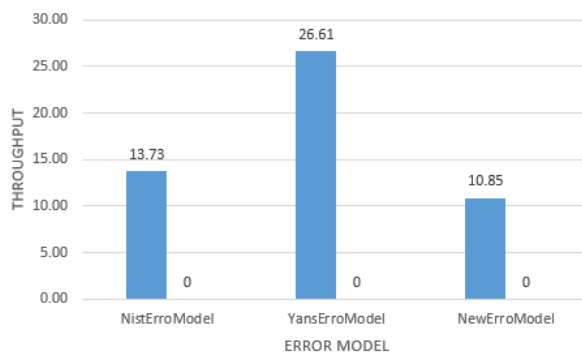


Figure 6. Comparison of error models

To the second case we are using a MANET with 30 nodes under a Random Way point mobility model.

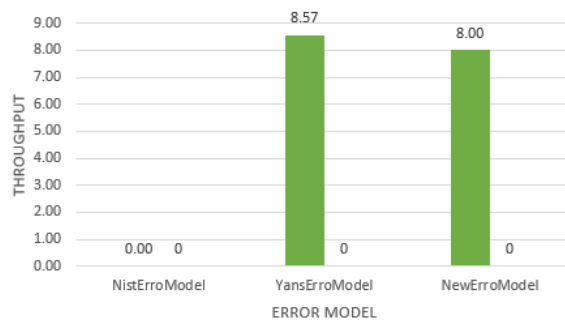


Figure 7. Comparison of error models

After analyzing the behavior, it was determined that in most cases the `YansErrorModel` model had a higher success rate. However, we consider that the variation of throughput depends a lot on the test scenario and the incident factors; such as: number of nodes, mobility model, speed of the nodes, test area, among others.

5. Conclusions

In this note, we have reviewed how to create a new model in NS-3. We provide a review of some simple steps. We show an example of the configuration for a new packet error model for 802.11p.

In most cases the `YansErrorModel` model had a higher success rate. This is because the `Yans` model is the simplest in terms of mathematical processes.

A determining factor when carrying out the simulations is the test scenario and its respective characteristics. Since the variation of any of these the success rate will change.

Supplementary Materials: In the previous note: "Deployment of NS3" you can find the virtual machine with which you can apply as indicated. In addition you can visit the following page: <https://perpapr2.lfurquiza.com/home> for more information.

Acknowledgments: The authors gratefully acknowledge the financial support provided by the Escuela Politécnica Nacional, for the development of the project PIJ-16-01 - "Modeling of the Packet Error Rate (PER) by including conditions of Peak-to-Average Power Ratio (PAPR) for Ad-Hoc. transmissions".

Bibliography

1. Abrate F., Vesco A., Scopigno R., An Analysis Packet Error Rate Model for WAVE Receivers, San Francisco, CA, USA, December, 2011.
2. 802.11 WIFI, brief, (2016). [Online]. Available: <https://es.ccm.net/contents/789-introduccion-a-Wi-Fi-802-11-o-Wi-Fi>
3. NS-3 Wi-Fi MODULE, (2017). [Online]. Available: <https://www.nsnam.org/docs/models/html/Wi-Fi-design.html#phy-layer-models>
4. New Modules NS-3, (2016). [Online]. Available: <https://www.nsnam.org/docs/manual/html/new-modules.html>
5. Maygua Marcillo, L. Y. (2017). Simulación de un mecanismo de control de potencia en aodv en función del número de vecinos usando cross layering en una MANET. 132 hojas. Quito : EPN. Available: <http://bibdigital.epn.edu.ec/bitstream/15000/19024/1/CD-8421.pdf>
6. 802.11a PHY LAYER, Geier J. (2014). [Online]. Available: <http://www.wi-fiplanet.com/tutorials/article.php/2109881/80211a-Physical-Layer-Revealed.htm>
7. New models, (2018). [Online]. Available: <https://www.nsnam.org/docs/manual/html/new-models.html>