Contents

List of Figures

List of Tables

Chapter 1

Tools and applications

In this chapter I will describe the tools and softwares I have utilized to carry out the simulations.

1.1 Radio Propagation Model

A Radio Propagation Model (RPM) is an empirical mathematical formulation used to model the propagation of radio waves as a function of frequency, distance, transmission power and other variables. Over the years various RPMs have been developed, some aiming at modelling a general situation, while other more useful in specific scenarios. For example, we can range from the more general free space model, where only distance and power are considered, to more complex models which account for shadowing, reflection, scattering and other multipath losses.

The authors of **6298165** classify the propagation models offered by the network simulator ns-3 in three different categories:

* **Abstract** propagation loss models, for example the Maximal Range model (also known as Unit Disk), which establishes that all transmissions within a certain range are all received without any loss
* **Deterministic** path loss models, such as Friis propagation model, which models quadratic path loss as it occurs in free space, and Two Ray Ground, which assume propagation via two rays: a direct (LOS) one, and the one reflected by the ground.
* **Stochastic** fading models such as the Nakagami model, which uses stochastic distributions to model path loss.

1.2 Obstacle Shadowing propagation loss model

The original thesis **ROM2017** used a deterministic RPM called Obstacle Shadowing propagation loss model presented in **5720204** and implemented by the authors of **Carpenter:2015:OMI:2756509.2756512**. This propagation model calculates the loss in signal strength due to the shadowing effect of obstacles such as buildings.

Other RPMs do not keep into conside

1.3 Network Simulator 3

Network Simulator 3 (ns-3) is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.

ns-3 development began in 2006 by a team lead by Tom Henderson, George Riley, Sally Floyd and Sumit Roy. Its first version was released on June 30, 2008.

ns-3 is the successor of ns-2, released in 1989. The fact that the former was built from scratch makes it impossible to have backward compatibility. In fact, ns-2 used oTCL scripting language to describe network topologies and C++ to write the core of the simulation. This choice was due to avoid the very time consuming C++ code recompilation, exploiting the interpreted language oTCL. ns-2 mixed the "fast to run, slow to change" C++ with the "slow to run, fast to change" oTCL language. Since compilation time was not an issue with modern computing capabilities, ns-3 developers chose to utilize exclusively C++ code (and optional Python bindings) to develop simulations.

1.3.1 Modules and module structure

ns-3 is composed of various modules, which are groups of classes, examples and tests each related to a certain feature. The components of a module work in a cohesive way in order to offer APIs to other modules and users. Some examples of built-in modules are:

* WiFi;
* AODV;
* CSMA.

The obstacle shadowing propagation loss model and the Fast Broadcast algorithm have been implemented as modules too.

Modules follow a prototypical structure in order to promote clarity and offer built-in documentation. Figure 1.1 shows the typical module structure.

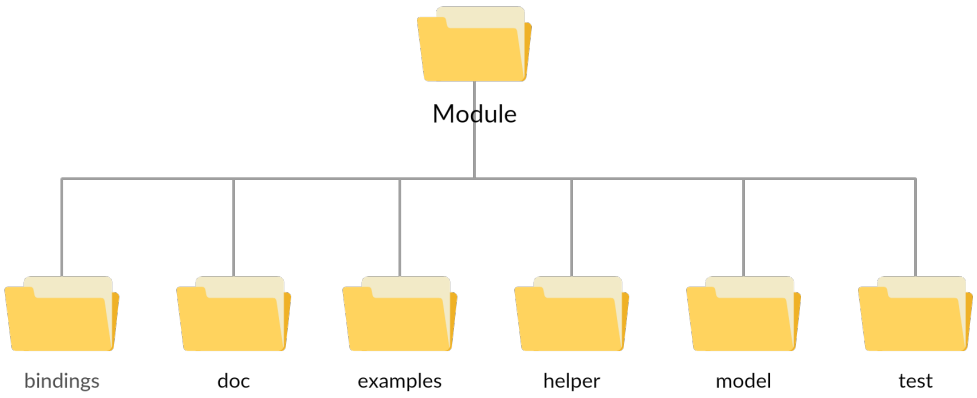

Figure 1.1: ns-3 module structure

The following directories can be found inside a module's root directory:

* **bindings:** Python bindings used to make the module's API compatible with Python;
* **doc:** documentation of the module;
* **examples:** examples and proof of concepts of what can be done using the module;
* **helper:** higher level APIs to make the module easier to use;
* **model:** headers and source files which implement the module's logic;
* **test:** test suite and test cases to test the module.

1.3.2 Key elements

The element at the base of ns-3 is called *node*, instance of `ns3::Node`. A node can be thought of as a shell of a computer. Various other elements can be added to nodes, such as:

* NetDevices (e.g. NICs), which enable nodes to communicate over *channels*;
* protocols;
* applications.

It is the last those which implement the logic of a simulation. For example, the `UdoEchoClientApplication` and `UdpEchoServerApplication` can be used to implement a client/server application which exchange and print the packets' content over the network. The Fast Broadcast protocol has been implemented as an application as well.

The *channels* model various type of transmission media, such as the wired and the wireless ones.

1.3.3 Structure of a simulation

A simulation can be implemented in many ways, but in most cases the following steps are executed:

* manage command line arguments (e.g. number of nodes to consider in the simulation, transmission range, etc.);
* initialize all the necessary fields in classes;
* create nodes;
* set up physical and MAC layers;
* set up link layer, routing protocols and addresses;
* configure and install applications on nodes;
* position nodes and (optionally) give them a mobility model;
* schedule user defined events, such as transmissions of packets;
* start the simulation;
* collect and manage output data.

1.3.4 NetAnim

Netanim is an offline animator tool based on the Qt toolkit. It collects an XML tracefile during the execution of a simulation and can be used to animate the simulation, analyzing packet transmissions and contents.

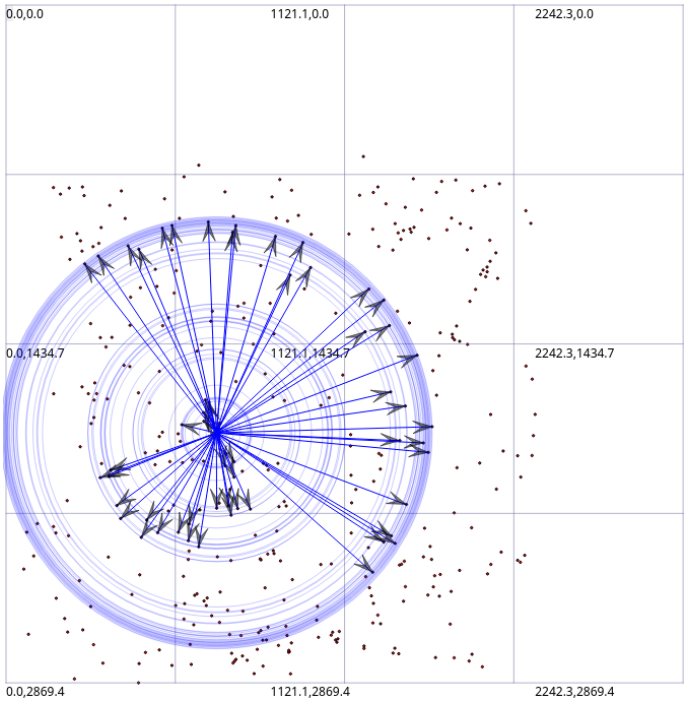

Figure 1.2: Packet transmission in NetAnim

1.4 Simulation of Urban MObility

Simulation of Urban MObility is an open-source road traffic simulation package. It is written in C++ and licensed under GPLv3.

It offers different tools to analyze and manage real maps from the urban mobility point of view, including pedestrian movement and various types of vehicles.

The original work **ROM2017** utilized SUMO to produce, starting from real maps obtained from OpenStreetMap (OSM), two files:

1. a `.poly` file using the SUMO tool Polyconvert. This file contains information about all the obstacles, such as buildings, useful for the Obstacle Shadowing module.

2. a `.ns2mobility` file using the SUMO tool TraceExporter. This file contains information about the vehicles and their positioning.

The process of generating these two files necessary for ns-3 simulations requires some intermediate steps. The full process is represented in figure 1.3.

The original work considered a only a distance of 25 meters between vehicles; this work considers various distances, ranging from 15 to 45 meters. Figure 1.4 shows the same scenario (Padua) with different distances between vehicles.

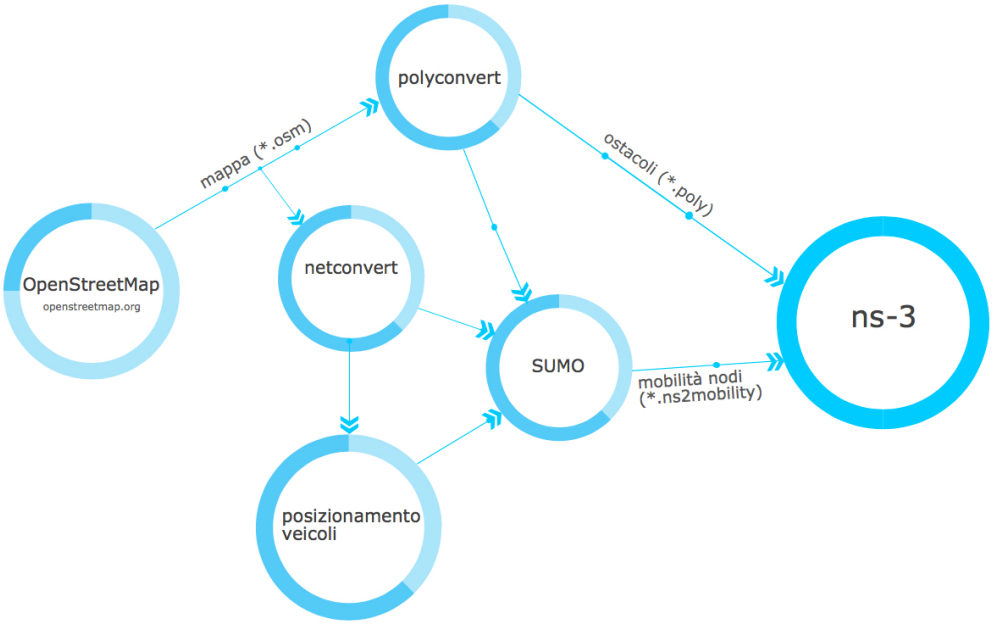

Figure 1.3: Steps to generate necessary files using SUMO (**ROM2017**)

Figure 1.4: Padua scenario with vehicle distance equals to 15 meters (top) and 45 meters (bottom). Vehicles painted yellow

Acronyms

RPM Radio Propagation Model. 1