Castadiva v0.92 Howto

Jorge Hortelano

1st July 2008

Contents

List of Figures

Part I

Installation

Chapter 1

Configuring the server

1.1 Requirements of Castadiva

Castadiva is a Java application developed using the Netbeans IDE. This programming language generates an executable compatible with almost all platforms and operative systems. The last Castadiva's release is generated with the java JDK 1.6, therefore is need the Java VM 1.6 to run it. Fortunately, *Castadiva* is free source and can be compiled to any other version of Java if needed. You can download the *Castadiva* application on `http://castadiva.sourceforge.net/`

As i said before, the core of *Castadiva* can run in any operative system, but some extra features (like the use of NFS folders) recommends the installation in a Linux system.

1.2 Installing Castadiva

To install the application in a computer, only is need a copy of the *Castadiva* files on the machine. No further installation is needed.

1.3 Configuring Castadiva

1.3.1 Configuring the NFS folder.

Castadiva share files with each router using the NFS shared folder system. It means that the core server must share a folder using the NFS protocol for *Castadiva*. This folder must be the same defined on Section 2.3.1 on the router configuration.

To export a NFS folder on a Linux system, we must edit the /etc/export file and add a line similar that showed on algorithm 1. For more information of how to use a NFS folder, read man pages.

Algorithm 1 Exporting a NFS folder.

```
/Castadiva *(rw,sync,no_subtree_check,no_root_squash)
```

Warning: Castadiva deletes all files and sub-directories on this folder before each simulation. Use a folder exclusively for Castadiva.

Chapter 2

Installing the routers

2.1 Installing the OpenWRT embedded system

The OpenWRT system is compatible with a large number of routers models. To confirm that your router is compatible with this system you can read the table of compatible hardware posted on the OpenWRT web page (`http://wiki.openwrt.org/TableOfHardware`).

OpenWRT has two development branches: the White Russian (also called OpenWRT 1.0) and the Kamikaze (OpenWRT 2.0). *Castadiva* has been developed using the White Russian branch.

The file need to install the OpenWRT system can be download from `http://downloads.openwrt.org/`, choosing the correct file to your computer. For example, for a White Russian for a Linksys router we must choose the *openwrt-wrt54g-squashfs.bin* file.

2.1.1 Quick installation

The best method to install the OpenWRT system on a router, is to use the firmware upgrade tool of the router (if available). Normally, in the administration panel of the router, we can select an option to upgrade to a new firmware. Figure 2.1 shows the administration panel of a Linksys router.

If is successful, this method will be enough to install the OpenWRT software. If not, read the next section.

2.1.2 Using a tftp client

If the quick installation method fails, ever can install it using a Tiny FTP. In our test we use the *atftp* application for Linux systems. The steps to follow to install the firmware are the next:

- To enter on the failsafe mode of the route, press the reset button for around 30 seconds. Then unplug a plug the router again while pressing the reset button.
- Connect the router to the computer's network (in the same network IP range).

Figure 2.1: Firmware upgrade of a Linksys router

- Enter on the tftp application.
  - Select the IP of the router: *connect 192.168.1.1*
  - Select the mode of the data: *mode octet*
  - Change the time between retries: *timeout 1*
  - Select a more verbose mode: *trace*
  - Upload the image: *put <path>/openwrt-wrt54g-squashfs.bin*
- Is probably that you must retry this method several times, until finally the software is upgrade. The tftp advise you when it happens.
- Finally, wait for a few minutes while the router install the new system, and reboot it.
- If you open a web browser and go to the URL of the router, you must to see the new interface showed at Figure 2.2.

This method is also useful if the firmware of the router has been erased and has nothing installed on it.

2.2 Configuring the OpenWRT embedded system

2.2.1 Using the OpenWRT web interface

The first step to configure the OpenWRT system is to use the web interface of this system. All this parameters also can be done using the commands line if you connect to the router by telnet. You must use the command line if you install the Kamikaze branch and has not installed the web interface on the router.

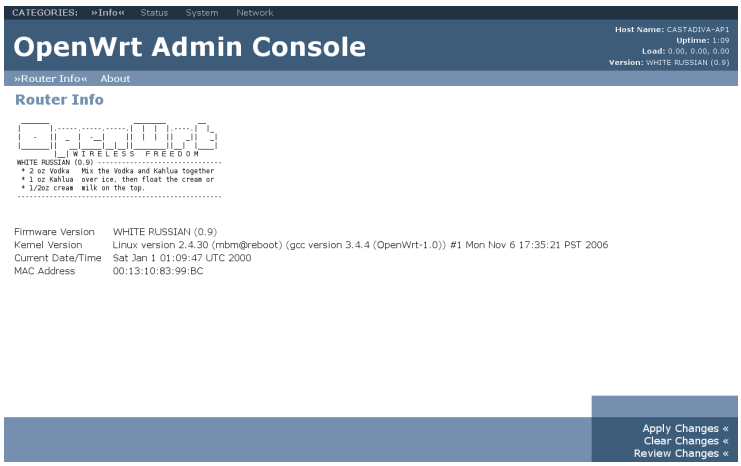

Figure 2.2: Starting administration web page of OpenWRT.

2.2.1.1 Changing the root password

The first thing you must do is set the root password of the OpenWRT system. This password will be used with the SSH connections and the web interface of the router.

The easier way to change the password is connect to the router and select the option on the web interface, inside the System menu.

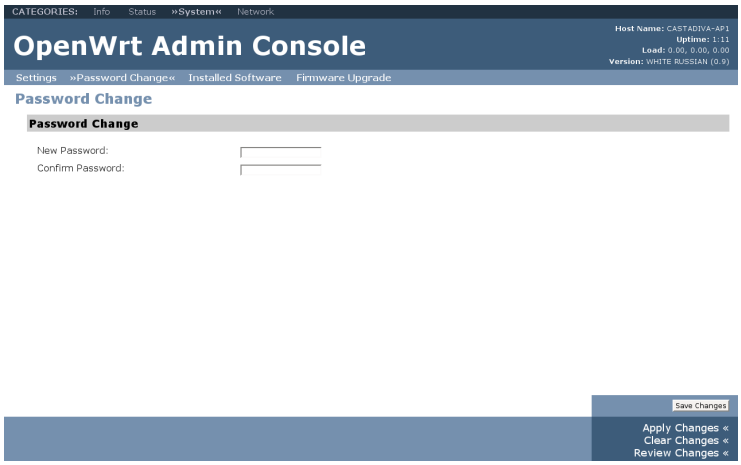

Figure 2.3: Changing the root password.

If you connect with telnet, you can also change the password using the "password" command, like others Linux systems. SSH will not be abailable until the root password is set.

2.2.1.2 Changing the Access Point's name

A good idea is set a name to the access point. It is not fundamental, but, when you connect to several routers by SSH can be a good idea distinguish each one by a name.

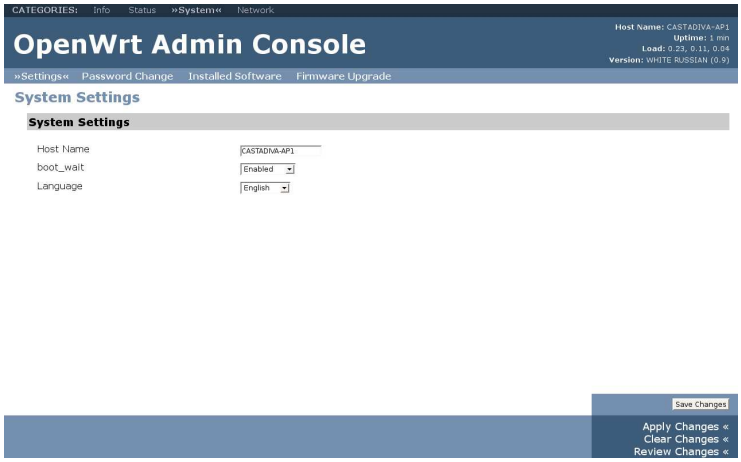

Figure 2.4: Changing the AP's name

You can change the router's name in the menu System, sub-menu settings. It is also a good idea to set the boot_wait option to enabled, because it will allows in the further to reinstall the operative system if there is any problem.

2.2.1.3 Configuring the network

The next step is configuring the wired network of the router. Feel free to select an IP range that meets yours requirements. E. g, you can put the IP 192.168.1.1 for router number one. It is important to establish a real gateway and a DNS server for the router. Later will install some packets and the router must access to Internet to download it.

I recommend that the last number of the IP match the name of the router for better administration.

Remember, when you apply this changes, if you have changed the IP of the router, change the URL of your browser to match the new IP, or you will be waiting for refreshing the web page indefinitly.

2.2.1.4 Configuring the wireless network

Now you must configure the wireless network. Feel free to select the IP addresses for each router, but remember that the IP range must be in different network that the wired network, avoiding that an accidental flow of packets uses the wired networks. The network connection must be in Ad-Hoc mode.

In my routers i assign the same last number for wired and wireless network, i. e. for the router number one, i configure the IP 192.168.1.1 for the wired network and the 192.168.2.1 for the wireless network. In this manual i refer to this number of IPs in my examples

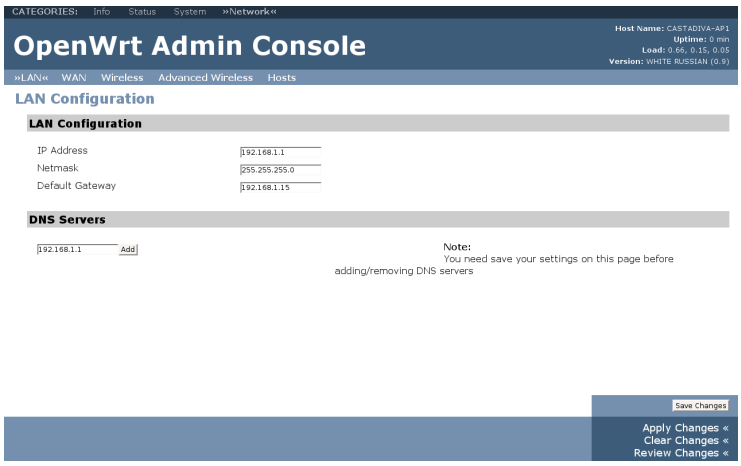

Figure 2.5: Configuring the wired network.

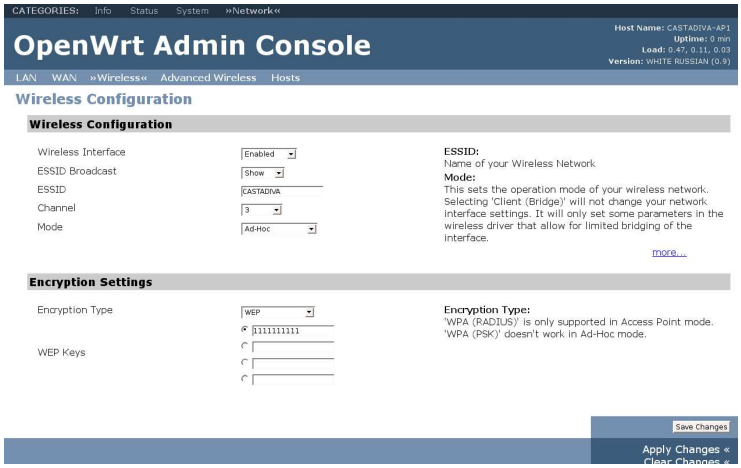

Figure 2.6: Configuring the wireless network.

As our routers can access to Internet, I recommend to put a WAP or WEP encryption to the network, to not share the Internet connection with strangers. WEP is not so secure, but some Linux systems has some problems to install the WAP encryption. The decision of what kind of encryption is used is yours.

2.2.2 Other configuration

The next steps are not mandatory but are recommended to facility the use of *Castadiva*.

2.2.2.1 Sharing the public keys between your router and your computer

Probably in the future you will access several times to each router by SSH. To speed up this procedure, i recommend to share the public key of your computer

with the routers.

- The first step is generate in your computer the public/private pair key if are not already created: *ssh-keygen -t dsa*
- Next copy the public key to each router: *scp ~/.ssh/id_dsa.pub root@-192.168.1.1:/tmp*
- Connect to your router by SSH and go to the Dropbear directory: *cd /etc/dropbear*
- Store the public key of your computer in your router: *cat /tmp/id_dsa.pub >> authorized_keys*
- If the file /etc/dropbear/authorized_keys not exist previously, remember to set the correct permissions: *chmod 0600 authorized_keys*

Now, each time you access by SSH from your computer to this router, you will does not need to type the password.

2.2.2.2 Configuring the OLSR routing protocol

If you want to use the OLSR routing protocol in your tests, you must configure it. *Castadiva* will install it for you, as will see on Section 2.3.1. The version of this protocol by default on OpenWRT White Russian is 0.4.10. You can also download a packet with the 0.5.5 version of this protocol for MIPS devices on the web page of *Castadiva*.

For configuring the OLSR protocol, you must edit the /etc/olsrd.con file (or /etc/olsrd-0.5.conf if you use the version 0.5.5 of the *Castadiva*'s webpage). In this file search for the line with the string "Interface" and write here the name of the wireless interface of your device (for the Linksys routers are the "eth1" but it changes on each model).

If you are testing the OLSR with a Mesh network the router can share its Internet connection. Change also the line Hna4 { @@HNA_IP@@ @@HNA_MASK@@ } on the device that access to Internet, selecting here the IP and netmask of the interface used to access to Internet.

You can also change others parameters of the OLSR like the Hello or TC Interval to adapt it to your test.

2.3 Using the Castadiva configuring tool to prepare the routers.

In the previous section you have installed the OpenWRT system and configure it, now, you are going to add some scripts used by *Castadiva* and install all OpenWRT packets needed by the test bed.

2.3.1 Only one click

All this process has been automatized. Now start *Castadiva* and select in the menu *Configuration* and the option *Install an OpenWRT AP*. A new window like the showed on Figure 2.7 will appears.

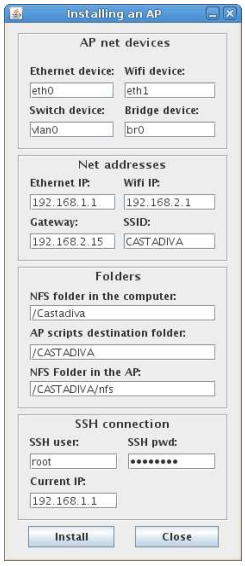

Figure 2.7: Installing an access point.

In this window, you must fill the interfaces' name of the device, the network configuration of the device and the SSH connection data. If the network configuration is different that the chosen on the web interface, *Castadiva* try to change to the indicated here the next time you reboot the device. The SSH data is used to connect to the router and install all the scripts and packets.

You also must fill the information for the NFS folder. The first field is the NFS folder on the core (computer), the second field is the destination folder on the router where *Castadiva* will install their scripts, and the third is where the NFS folder will be mounted on the router.

When you press the *Install* button, *Castadiva* generates a script on the folder selected before. This script configures the network of *Castadiva* (for example, disable the bridge mode not desired for the *Castadiva* environment) and executes an instruction to mount the NFS folder. To automatize the execution of this script, *Castadiva* also generates a link in /etc/init.d/ on the router to force the execution of the script every time the router starts.

The Install action also force the OpenWRT to install all needed packets if are not already installed. By default install the packets *kmod-nfs, iptables-extra, tcpdump, libgcc, libpthread* and *olsrd*. The packets to install are defined on the file *<path of Castadiva>/configuration/packets.txt* of the *Castadiva*'s core application. All changes on this file, changes the list of packets to be installed by *Castadiva*.

2.3.2 Checking that all is correctly configured

When the installation is finished (you must wait for a few minutes because *Castadiva* does not advise you when it finish), is a good idea to check if all packets and scripts are installed. First reboot the router to load the new configuration and:

- Check if all packets are installed. It will fail if the router has no Internet access. If it is the case, go to Section 2.2.1.3.
- Check if the scripts are generated. A script called *Castadiva.sh* must be generated on the folder defined in the installation window, in the field *Ap scripts destination folder*.
- Check if the bridge of the router is unmounted and the interfaces of the wired and wireless network are defined.
- Check if the NFS folder is mounted. If it fails, run manually the script *Castadiva.sh* to see if there is any errors.
- If you miss some configuration and can not access to the router, you can access to the failsafe mode on the router pressing the reset button before plug in it, or reinstall the OpenWRT as showed on Section 2.1.2.

If there is no error, the router is ready to become one node of *Castadiva*.

Part II

Using Castadiva

Chapter 3

Understanding Castadiva

3.1 Architecture

Castadiva is a test-bed designed to deal with the development and the performance evaluation of protocols and applications for MANETs. The test-bed relies on an actual IEEE 802.11 wireless network between nodes for testing purposes. *Castadiva* is composed by a server that runs the main application, several wireless nodes, two different networks and an application that coordinate all devices.

Castadiva's server executes the application and configures the network devices. Concerning the wireless nodes used, they can be any sort of computing device, like a laptop, a PDA or a wireless router. In our prototype, each node is a Linksys router to minimize costs. The main requirement for a node is that it must have a Linux/Unix operating system installed, and two network cards: an Ethernet card and an IEEE 802.11 card. If the node is a wireless router, the OpenWRT [1] kernel is an optimum solution. OpenWRT is an open source operating system available for a wide range of router manufacturers. This embedded Linux system natively offers SSH connections, along with the possibility of running shell scripts. Moreover, a programmer can develop its own application in a standard Linux distribution and export it to this operating system. In our case, we developed some applications in C for traffic generation and control purposes.

Figure 3.1 shows a schema of *Castadiva*'s components. The main application, developed in Java, controls all devices and manages the links among them according to a pre-defined network topology. It also manages traffic generation between pairs of nodes. Since the controlling application also requires communicating with nodes to send control packets, *Castadiva* combines two different networks: the coordination network (wired), that connects the *Castadiva* core with the wireless nodes, and the wireless network, where actual tests run.

The coordination network is a wired network that connects *Castadiva*'s core server with the wireless nodes. This network allows the main application to send configuration messages to all the nodes without creating any interference within the wireless network itself. It is based on Fast-Ethernet technology, to avoid

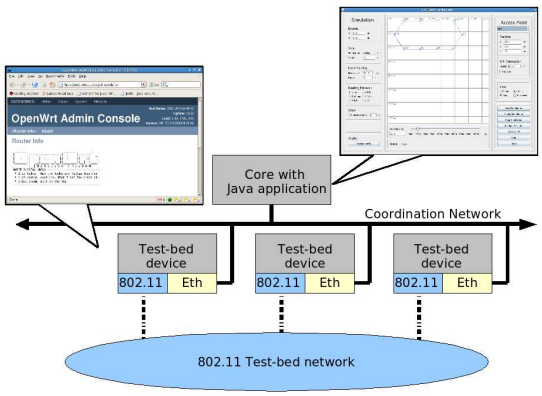

Figure 3.1: Schema of *Castadiva*'s components.

large latency. Basically, the coordination network requires a switch connected to the main server and to all nodes. Through this network the main application sends instructions to nodes, allowing them to re-configure so as to create the desired network topology, and also to run lightweight traffic-generating applications available on each wireless node. For communication purposes, we rely on the SSH protocol to send instructions through this network. We use the JCraft [2] java SSH library for this purpose. Using a fast network means that all nodes participating in a test will start at about the same time, avoiding significant latency effects and maximizing result accuracy.

The wireless network is composed by *Castadiva*'s wireless nodes, and the topology of this network is defined by the GUI of *Castadiva*, so that it can change at runtime. Nodes communicate in ad hoc mode using IEEE 802.11g technology.

Figure 3.2: *Castadiva*'s physical network.

Castadiva's core has two main functions: (a) to allow a user to interact with the system so as to define all the test parameters required and (b) to coordinate the wireless nodes during an experiment. By using *Castadiva*'s GUI a user can control all of *Castadiva*'s functionality, defining the network topology and the

traffic flow among nodes. *Castadiva* allows fixing the scenario area where nodes will be deployed. When selecting a node, its location is highlighted and it can be changed according to the desired network topology. When all nodes are deployed the user can press the *Simulate* button, and each physical node will be re-programmed so as to enforce the chosen network topology.

Figure 3.2 shows our test-bed. One switch connects *Castadiva*'s server with all the wireless nodes for coordination purposes. On the center of the picture the group of wireless nodes being used are shown. It consists of eleven Linksys routers (models WRT56G and WRT56GL). The wireless ad hoc network conformed by these nodes is the one used in *Castadiva*'s testbed experiments. The experiments presented in this work required extending *Castadiva* to support traffic injected from outside applications. In particular, laptops to go through our MANET, allowing us to assess the performance of video conference sessions as experienced by users.

Chapter 4

Features

4.1 Graphical user interface

Castadiva's core element, a Java application running at the server, includes all the control functions required for testbed experimentation. It is responsible for network topology maintenance, traffic control, as well as result calculation and presentation. A user can define the characteristics of wireless nodes. Each node is deployed at a specific position in a simulated area as chosen by the user. Once the topology is defined, *Castadiva* must configure the wireless nodes according to that topology. The application communicates with each node through SSH connections to send the required instructions. The traffic flow between nodes and the routing protocol used are also set through this technique. When all the experiments are finished, *Castadiva*'s core must calculate the result statistics for the experiment by gathering all the data obtained, and finally show these results to the user.

Castadiva's main application was created using Java's Swing library. We consider that it is a good solution for visual design since most basic components are already created, and can be easily modified by the programmer.

Castadiva is designed to be a test-bed where network scenarios and traffic between nodes are generated so as to resemble a real MANET. Therefore, it is expected to be an easy and useful tool for the study of MANETs.

To start a new experiment we only need to define the network topology in the corresponding window and then define the traffic flow and the routing protocol used. By pressing the start button tests begin, and *Castadiva* returns the test results automatically at the end of the simulation. We now offer more details about *Castadiva* offered services:

Adding Nodes To The Testbed

Before starting an experiment the user needs to define the number of participating nodes, along with their configuration. Such information allows *Castadiva* to access nodes and manipulate them to generate a scenario. Figure 4.1 shows an example of the definition of a node in the system.

All the information is defined automatically when the user wishes to add a new one, though it can be changed by the user or can be read from a file.

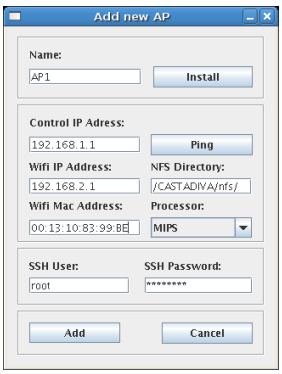

Figure 4.1: Node configuration interface.

An internal identifier is required to distinguish a node from others in *Castadiva*'s framework. Such identifier is then referenced when defining the network topology and data connections. The remaining parameters will be used by *Castadiva*'s main application to connect nodes among themselves and with the main server. The MAC address is required for *Castadiva* to enforce topology changes.

All the executable files and the scripts are stored in an NFS directory that is accessible by all nodes. This way *Castadiva* makes storage capacity independent of wireless nodes' memory.

Castadiva relies on its own tools to generate traffic between nodes. Such tools are run on each node, and must be compiled for all types of CPU used. Currently, tools are compiled for MIPS and Intel processors, though the list can be easily augmented.

The SSH user and password fields are used by the main application to connect to each individual router and submit commands. Also, a Ping button was included to allow testing the connectivity between the server and routers.

Ad Hoc Network Scenario Generation

Once all the nodes are defined, they can be distributed to conform a scenario. *Castadiva* supports both manual and random topology generation, and the scenario is set through *Castadiva*'s blackboard. The blackboard is a representation of a virtual environment where nodes are located. Nodes are differentiated through different colors and labels. If the appropriate option is selected, the radio communication range is also shown through a circle of the same colour.

Figure 4.2 shows the topology generation environment. We can see ten nodes located in a scenario of 1500 x 1500 meters (scenarios bounds are marked with a darker line).

At the right hand we may edit node properties, such as position and signal range. *Castadiva* also can activate or deactivate the RTS/CTS 802.11 option of each node. The group of buttons appearing below allow starting a new test, stopping it and rebooting nodes to reset all values.

At the left hand, *Castadiva* offers scenario option editing. We can define the scenario bounds, the test time, node mobility and the routing protocol used.

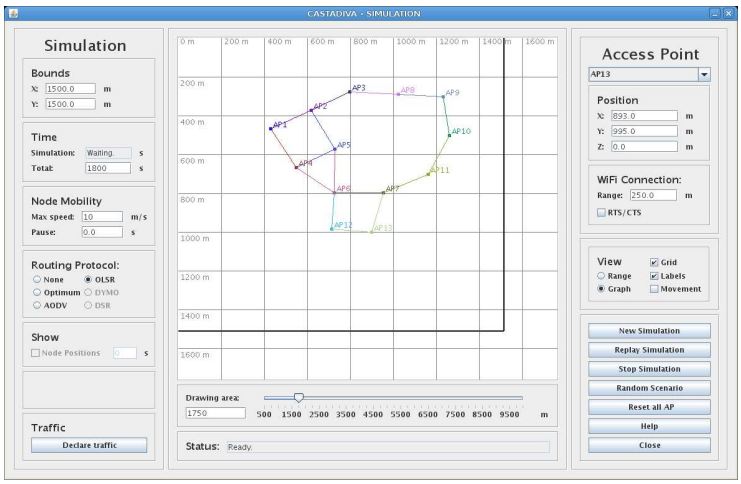

Figure 4.2: Scenario definition with *Castadiva*.

The *Declare traffic* button allows setting traffic, as shown below, and the stop button halts it.

A status bar provides general information to inform the user about what is being done, and the horizontal scrolls allows zooming in and out. Finally, the user may alternate between two different views: radio ranges or a graph. Every edge of the graph represents an IEEE 802.11 link connection, which is a more intuitive view.

Mobility in Castadiva. It is important to point out the speed and pause option of the Scenario Window. If a user picks a value greater than zero in the speed option, each node acquires a random movement with a speed between zero and the inserted value. When a node arrives to a destination point, it waits for a selected pause time and then select a new random destination point to move to. This behavior is similar to the one provided by the "setdest" tool embedded in the ns-2 simulator.

For the representation of the mobility, *Castadiva* generates all node movements required for the simulation before it starts. It then calculates the visibility range for each node every second. The obtained visibility is translated to Iptables rules and written to one file for each node. These files will be loaded on each access point through NFS when the simulation starts. Figure 4.3 shows the file loaded by AP1. We can see in this figure different Iptables rules inserted among *sleep* rules. The sleep time allows to enforce Iptables rules at the appropriate times. So each rule is loaded only when the emulation requires a node to change its connectivity state towards other node. E. g. Algorithm 2 shows what is the behaviour of *Castadiva* when a node (with MAC 00:14:BF:3C:39:EC) goes out of range at second 15 and comes back into range at second 35.

Castadiva also allows an user to see all node positions at a certain instant of time. When a simulation finishes, the user can activate the *Show* option and pick an instant of time. Immediately *Castadiva* shows the network topology at that time. This option is useful to do a later evaluation of the changes occurred in a network topology when mobility was activated.

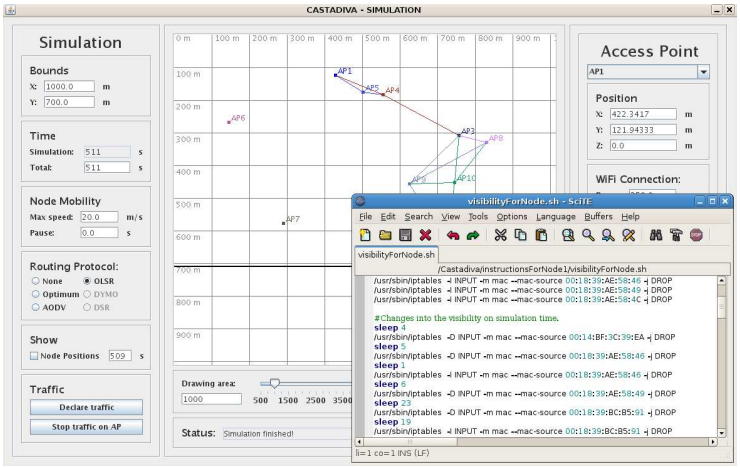

Figure 4.3: Mobility implementation.

Algorithm 2 Iptables rules for a simulation between two nodes.

```
sleep 15
iptables -I INPUT -m mac –mac-source 00:14:BF:3C:39:EC -j DROP
sleep 20
iptables -D INPUT -m mac –mac-source 00:14:BF:3C:39:EC -j DROP
```

Network Traffic Declaration

Castadiva's traffic generation tool allows defining different types of traffic flows between pairs of nodes. With that purpose *Castadiva* provides a table where each row defines a connection. Traffic parameters for each connection can be set depending on the type of protocol selected, and invalid values are marked with red. Examples of parameters are: *packet size*, *packets per second*, *start time*, *end time* and *maximum number of packets sent*. Figure 4.4 shows a usage example of this tool, where rows define seven traffic connections. It contains some helpful buttons that allow making row operations (delete, order by starting time, or copy). Traffic settings are exportable to NS-2 format also.

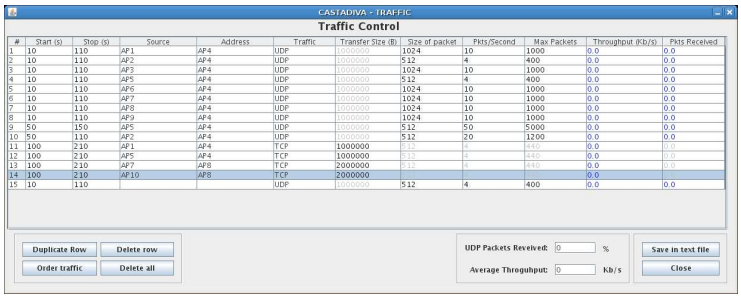

Figure 4.4: Traffic declaration tool.

When an experiment finishes, *Castadiva* fills in this table with results, including throughput. *Castadiva* also shows the percentage of UDP packets received

and the average throughput value in the case of TCP traffic.

Random test generator

Sometimes it is useful to automate the testbed evaluation process varying different parameters. With that purpose *Castadiva* includes functionality to generate random tests, where a user can define traffic and automatically test with different number of nodes and randomly-generated network topologies. This is achieved through the *Random test window* shown in Figure 4.5.

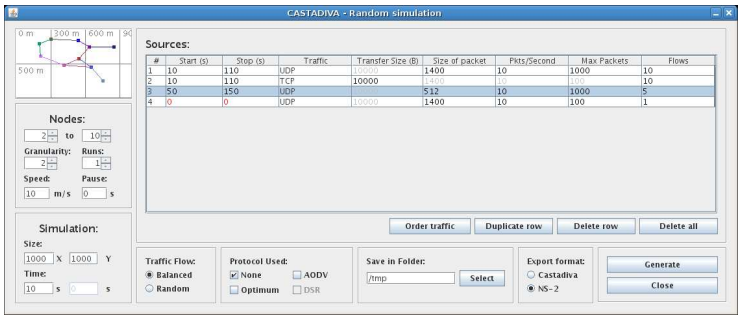

Figure 4.5: Random test window.

The user must specify the bounds of the scenario and the routing protocol used. The minimum and maximum number of nodes for testing must also be defined, along with the increase granularity. (e. g., with a node interval between 4 and 10 nodes and a granularity of 2, *Castadiva* executes four tests with 4, 6, 8, and 10 nodes). *Castadiva* also allows to specify how many times each test will be repeated.

At the top left the current scenario generated is displayed, though it can not be modified. Again, all the tests can be stored in either *Castadiva*'s or NS-2's format.

Compatibility with the ns-2 simulator:

Castadiva can also import/export scenarios to/from ns-2, making it compatible with the most widely used MANET simulator. This characteristic offers the possibility of comparing results and reaching more meaningful conclusions. You can export or import to ns-2 selecting *Export* or *Import* on the *Application* menu.

Castadiva extensions for external traffic injection.

Such functionality allows external nodes to generate real traffic of any kind and redirect it to specific nodes of *Castadiva*. For this reason *Castadiva* also incorporates an extra functionality that allows attaching a laptop or a computer to a node. Figure 4.6 shows an example of this functionality.

Such functionality allows the use of laptops to generate any flow of traffic and redirect it to specific nodes of *Castadiva*. For example, you can use real applications like Ekiga or Skype to launch a video-conference and study the behaviour of H.323, SIP and video streaming protocols in MANETs.

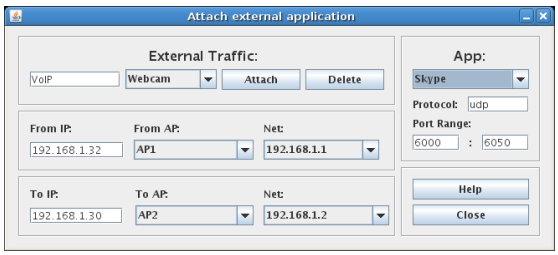

Figure 4.6: External traffic declaration.

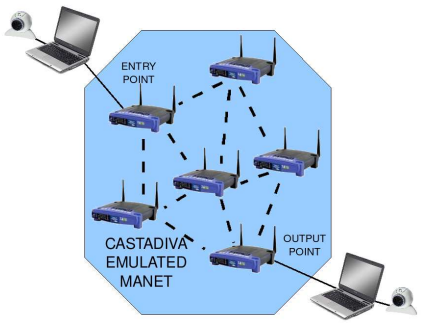

Figure 4.7: Example of how to add external traffic injection.

Figure 4.7 shows an example of two laptops connected to *Castadiva*. Both laptops have a webcam and run the Ekiga application to generate a videocall. *Castadiva* redirects all traffic related to this video-conference through the emulated MANET, from the entry to the output point.

4.2 Other secrets of Castadiva

In this section i explain other characteristics of *Castadiva* not described before.

4.2.1 Subfolders on the NFS

Each time *Castadiva* starts a simulation, it deletes all old files and folders on the NFS server, to avoid use old data on the new test. But if you want to share other applications with you router, you can also use this NFS folder. *Castadiva* is programmed to not delete a folder called *bin* a folder called *notDelete* and any file that contains the string *Flow* on its name like *TrafficFlowClient* used on traffic generation. Feel free to use this folders to your purpose.

4.2.2 Adding default configuration for the nodes of Castadiva

If you are thinking to generate lost of different tests with *Castadiva*, it will be a little annoying to fill each time the *Add new AP* window when adding a new

node. To simplify it, you can change the file *<Castadiva's path>/configuration-/aps.txt* on the core computer. Each line on this file will be used each time on the *Add new AP* window.

4.2.3 Adding default applications to the external traffic window

As described in previous section, can be annoying to fill each time some fields of *Castadiva*. For *Attach External Application* window you can set a group of default applications. Change the file *<Castadiva's path>/configuration/applications.txt* filling all needed applications in your simulation.

4.3 Examples of emulation with Castadiva

4.3.1 My first test with Castadiva

This first example teach how to use the basics concepts of *Castadiva*. How to add nodes to the simulation and how generate traffic flow between nodes.

4.3.1.1 Adding nodes to the simulation

As explained in Section 4.1 we must to insert nodes to be used on the emulation. In this example we only need three nodes.

Select on *Configuration* menu the *Add Access Point* option. Fill the access point name field as "AP1". Fill all other fields if is needed. Repeat this step for nodes 2 and 3 changing the name to "AP2" and "AP3".

4.3.1.2 Generate the network topology

Now select on *Simulation* menu the *Generate Simulation* option. It will open the *Simulation* window. Now, in the right hand we can select one of these three nodes.

Select the node "AP1" and click on the blackboard (the square on the center of the window). Repeat this procedure for nodes "AP2" and "AP3" until have some similar to the showed on Figure 4.8.

Now change the simulation time to a value of at least 60 seconds.

4.3.1.3 Declare traffic

Now you must add some traffic flow to our example. In the *Simulation* window click on the *Declare Traffic* button to open the *Traffic* window.

In this window select on the first line of the table and change the *Source* and *Address* columns to the AP1 and AP3 options. You can change other parameters of the traffic, but for this example an UDP traffic with the default values will be enough.

When a line is completely filled, a new one appears to add an extra traffic flow.

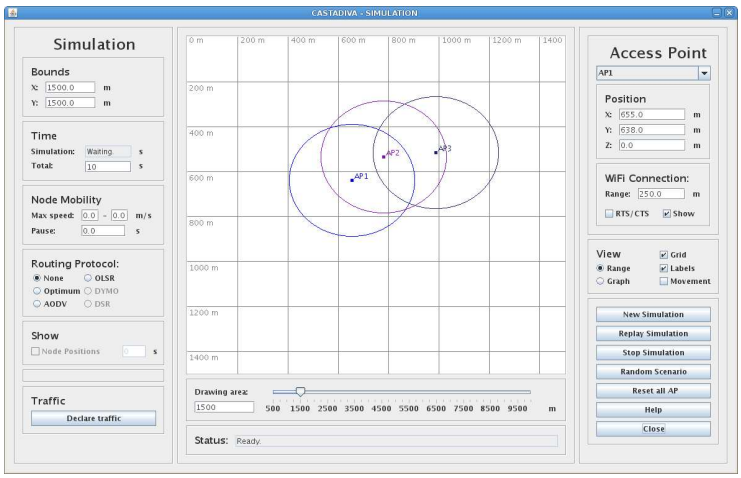

Figure 4.8: A simple test.

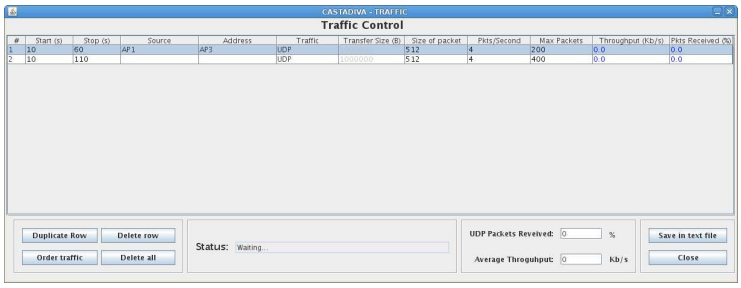

Figure 4.9: Adding traffic to the first example.

4.3.1.4 Simulate

Now you can start the simulation. Return to the *Simulation* window and press the *New Simulation* button. You can see:

- The simulation status (on the bottom of the window) changes the message.
- The simulation's time counting. If the final time is greater than that you have been selected before, is because the traffic time interval selected on the *Declare Traffic* window is greater than the simulation time. *Castadiva* changes it value before starting.
- If you go to the NFS folder you can see that a new folder for each node have appears.
- If you connect by SSH to node AP1, can see that are new Iptables rules blockading the MAC of AP3.

When the simulation finish. You can see the results obtained: the *Declare Traffic* window opens if is closed with the last two columns filled. In this example the result obtained must be 0 if you have not selected a routing protocol. Replay the test selecting a routing protocol and this result will change.

4.3.2 A test with laptops and video traffic using Ekiga

This example describes how to use video traffic using laptops like other *Castadiva*'s nodes.

4.3.2.1 Adding nodes to the simulation

Add different routers as explained in Example 4.3.1. Also add two laptops like a nodes using the same steps. Remember that this laptops must use the NFS folder, must be in the same network and have all the same configuration like other node. This laptops also need the Ekiga application and a webcam installed on it to generate video traffic.

4.3.2.2 Generate the network topology

As in other tests, deploy the nodes on the network topology using the mouse. You only can differentiate here the laptops by the name of the node. *Castadiva* does not distinguish the different devices used.

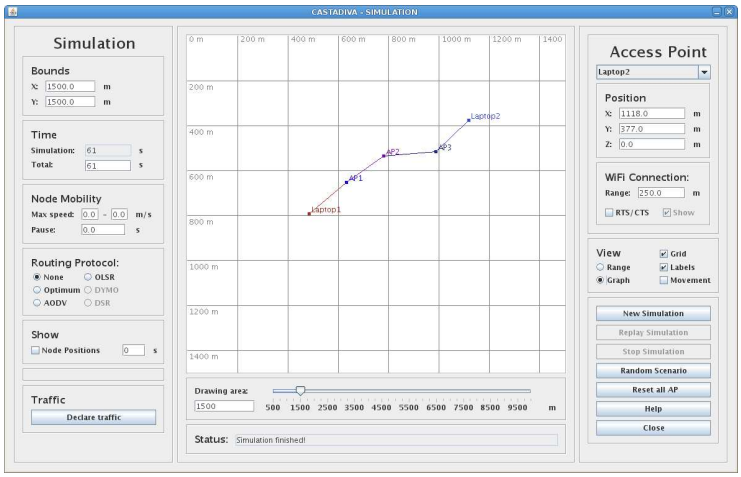

Figure 4.10: Topology of a test using laptops.

Figure 4.10 shows an example of topology.

4.3.2.3 Declare traffic

You do not need to declare traffic on Castadiva. You simply start the Ekiga application on each laptop and make a videocall.

Note: The External Traffic Window is used only if the laptops are not used on the network, to add extra functionality to a router of Castadiva. In this example is not needed because the laptops are part of the network. This option will be discussed on Example 4.3.3.

4.3.2.4 Simulate

You can start the simulation. Remember that the traffic is external to *Castadiva*, then for the emulator is impossible to obtain the results. You need other

application as tcpdump to obtain some results.

4.3.3 A test with using Ekiga as external traffic

This example describes how to use video traffic without using laptops as *Castadiva*'s nodes.

4.3.3.1 Adding nodes to the simulation and Generate the network topology

For this test, repeat the topology of Example 4.3.1.

4.3.3.2 Declare traffic

Now, start the Ekiga application on the laptops. As the laptops are not in the topology, you must attach the traffic flow to the routers of the topology. For this, open the *External Traffic* window in the *Configuration* menu window.

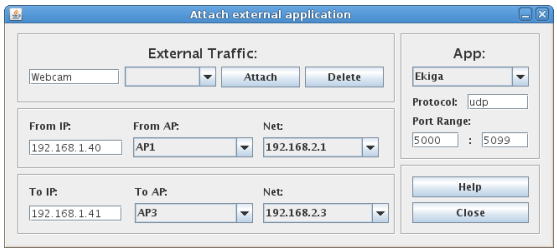

Figure 4.11: Attach external traffic.

Figure 4.11 shows an example of how to fill all fields. In the *From IP* field fill the IP of one of the laptops. Next select an AP to attach this traffic flow and select the wired IP on the *Net* field. Repeat this procedure for the other laptop. This window sets the parameters to allow *Castadiva* to configure the routers to allow the traffic redirect. To confirm this traffic flow, press the *Attach* button.

As the laptops are not inserted on *Castadiva*, you must configure it manually: change the routing table to redirect traffic through the access points. To reach the other laptop, must use one of the routers of *Castadiva*. It is something like "route add -host <IP other laptop> gw <IP router> dev eth0".

When the simulation starts, the traffic will be redirected through the selected nodes.

Bibliography

[1] OpenWRT, wireless freedom. Available at: http://openwrt.org.

[2] A. Yamanaka and JCraft Inc. Jsch, the java secure channel. Available at: http://www.jcraft.com/jsch/.