Performance Evaluation of a Mobile Ad hoc Network Test-bed Architecture*

Jorge Hortelano, Marga Nácher, Juan-Carlos Cano
Carlos Calafate, Pietro Manzoni

Department of Computer Engineering, Technical University of Valencia
Camino de Vera s/n, 46071 Valencia, SPAIN.

*This work was partially supported by the Ministerio de Educación y Ciencia, Spain, under Grant TIN2005-07105-C02-01 and by the Generalitat Valenciana, Ayudas complementarias para proyectos de I+D+i, Spain, under Grant ACOMP07/237.

Abstract

The widespread use of the 802.11 technology offers new possibilities to extend network connectivity outside wired network. The development of new protocols and applications requires to be tested before their deployment in a real environment. The most common solution consists of using computer-aided simulators. However, it becomes more and more important to migrate the test platform to real test-beds solutions. In this work we present Castadiva, our test-bed architecture based on low-cost off-the-shelf devices and free software to test protocols developed for Mobile Ad Hoc Networks or MANETs. Castadiva is completely compatible with the file format used by the ns-2 simulator, so enabling a fair comparison in a in a simple and straightforward manner. We compare Castadiva with respect to the ns-2 network simulator, and demonstrate that the proposed solution is a candidate test-bed solution to evaluate software solutions and protocols in MANETs.

1 Introduction.

Mobile ad-hoc networks (MANET) [1] are wireless networks with no fixed infrastructure. Nodes belonging to a MANET can either be end-points of a data interchange or can act as routers when the two end-points are not directly within their radio range. The growing research efforts focusing on this new technology requires from the availability of tools that allow researchers to evaluate their proposals. Testing and evaluating any of the proposed protocols for MANETs is a mandatory request to guarantee its success in any real world application. Researchers have two different options for testing their proposals, either simulation tools or real test-beds platform.

Among the available networks simulators for MANETs, ns-2 [2] is the most popular one among the research community and the academia. ns-2 is a discrete event network simulator, which can be easily extended and have a great variety of online documentation. ns-2 has been popularly used in the evaluation and analysis of routing protocols proposed for MANETs. Despite the flexibility of using a simulation tool, their use also presents some important disadvantages. Just as an example, lacking realistic models for node mobility and radio propagation, there could be significant differences between the results achieved in real implementation test-beds compared to simulation. A test-bed platform permits to obtain rigorous, transparent and replicable testing of scientific theories and protocols. With respect to simulation tools, they are usually based on expensive infrastructure. So, to become affordable tools reducing their cost is a mandatory issue.

In this work we present Castadiva, which is our test-bed architecture that facilitates mak-
ing test-bed experiments; it relies on low-cost, of the-shelf devices combined with the power of a Linux platform. Castadi is allows generating network topologies, exporting them to real devices and obtaining the test results. It can also generate different types of traffic between nodes, and offers support for some well-known ad-hoc routing protocols. It relies on a cheap architecture that includes two different networks: a wired network, called connection network, that connects the core with a group of wireless nodes, and a wireless network where the actual test-bed experiments are made. We developed a group of tools for administration purposes, with a friendly user interface design to help the user to define the scenario of the network and the desired traffic connections between MANET nodes. All of these tools were developed with open source software.

Since all tests rely on wireless communication between autonomous devices, they approach the real behavior of a MANET. The main application merely coordinates all nodes and enforces topology changes, being all the tests executed in the wireless nodes alone. Our application is completely compatible with the ns-2 file format enabling a comparison in a simple and straightforward manner.

We selected a wide range of both static and dynamic MANET’s scenarios and compare the obtained results in Castadi with respect to those obtained using ns-2. We confirmed that Castadi is able to obtain confident results while using real wireless off-the-shelf devices.

The rest of this paper is organized as follows: Section 2 describes Castadi’s architecture. Section 3 presents the implementation details and, in Section 4, we compare this tool with the NS-2 simulator. Finally, in Section 5 we present our concluding remarks.

2 Architectural Overview

Castadi is a test-bed developed to evaluate and analyze protocols and applications for MANETs. The test-bed relies on an actual wireless network between nodes for testing purposes.

The main application, developed in Java, controls all devices and manages the links among them according to a pre-defined network topology. Also, it manages traffic generation between pairs of nodes. Since the controlling application also requires communicating with nodes to send control packets, Castadi combines two different networks: the coordination network (wired), that connects the Castadi core with the wireless nodes, and the wireless network, where actual tests are run. Figure 1 shows a schema of Castadi’s components.

Figure 1: Schema of Castadi’s components.

The coordination network is a wired network that connects Castadi’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, avoiding large latency. Basically, this 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 reconfigure so as to create the desired network topology, and also to run small traffic-generating applications available on each wireless node. For communication purposes, we rely on the SSH protocol to send instructions through this network. Using a fast network means that all nodes will start
an experiment at about the same time, avoiding significant latency and maximizing result accuracy.

The wireless network is composed of 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.

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 button Simulate, and each physical node will be re-programmed so as to enforce the chosen network topology. Figure 2 shows how Castadiva allows a user to interact with the network. We describe the whole functionality offered by Castadiva’s GUI in Section 3.2.1.

Castadiva’s server executes the application and configures the network devices. It consists of a Pentium IV with 1 GBs of RAM memory, and has a Linux Debian Etch distribution installed.

Concerning the wireless nodes used, they can be any sort of computing device, like a laptop, a PDA or a wireless router. The main requirement for a node is that it must have a Linux/Unix operative system installed, and two network cards: an Ethernet card and an IEEE 802.11 card. If the node is a wireless router, the OpenWRT [3] kernel is a good 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 operative system. In our case, we developed some applications in C for traffic generation/control purposes.

![Figure 3: Castadiva’s physical network.](image)

Figure 3 shows our test-bed. One switch connects CASTADIVA’s server with all the wireless nodes for coordination purposes. On the right hand of the picture the group of wireless nodes being used are shown. It consists of ten Linksys routers (models WRT54G and WRT54GL) and a Buffalo router (model WZR-RS-G54). The wireless ad hoc network conformed by these nodes is the one used in Castadiva’s testbed experiments.

3 Castadiva’s Implementation details

In this section we detail the requirements of Castadiva on the server and on the wireless nodes. We describe the software tools we have developed to connect all the wireless nodes with the server, and how Castadiva allows making connections among them. We also explain the process of designing network topologies by using the Scenario Generation tool, an interactive and user-friendly interface that allows defining the network’s scenario and the desired traffic connections among nodes.

Castadiva requires some libraries and services to operate. The requirements of Castadiva are different for the server and the wireless nodes. The server must be a standard Linux-based system and must have a Java Virtual Machine, an SSH client and an NFS server.
installed. Each node must be a Linux based system with an SSH server and an NFS client; besides, it must include the libgc library and have the **Iptables** toolset installed.

The connection between Castadia’s core element (server) and each node is made using both SSH and NFS connections. On Castadia’s server, the user interacts with the application by defining the network topology, the traffic and selecting the desired routing protocol. Then, through SSH, the application sends a starting instruction to each node through the coordination network (wired). Wireless nodes achieve coordination among themselves by executing the required binaries, which are stored into a server folder shared through NFS. This is an easy way to spread instructions to all nodes, and it solves storage limitation problems on nodes. When tests start, a group of files with the node configuration instructions are created and stored into Castadia’s server by again relying on the NFS filesystem. We find that Ethernet connections are fast enough to export these files to the routers without significant delays.

The main application parses the results, obtaining the different testbed statistics. Finally, the application displays results to the user.

The application was developed using the Java programming language and the BASH scripting language. To make SSH connections through Java we use JSch [4], by JCraft. It is required to coordinate all the nodes during experiments. Castadia’s implementation can be divided into two parts: the main application and the light-weight applications running on wireless nodes. The latter are the focus of the next section.

### 3.1 Wireless nodes’ software

Each node has a set of requirements that must be met for successful operation: a Linux-based operative system, a set of special-purpose scripts, some specific applications and connectivity to Castadia’s server.

The operating system installed on each router is OpenWRT. OpenWRT allows executing BASH scripts natively; besides, it includes Dropbear, a simple SSH server used to receive instructions from Castadia’s server. Concerning the set of Castadia’s scripts, they are generated automatically by Castadia’s main application. Their purpose is to configure the wireless network topology. Each node makes use of three applications: Iptables, TcpFlow, and UdpFlow. The first one is open source and exist in Linux distributions, while the other

![Figure 2: A view of Castadia’s GUI.](image)
two were developed by us.

Network topology configuration is made through the Iptables [5] tool. According to the selected topology, Iptables allows us to dynamically break the network links between pairs of nodes. This tool exists for all Linux distributions, including the OpenWRT embedded system. To generate traffic we create the UdpFlow and TcpFlow tools. Both tools are designed to create a traffic flow between two nodes; each tool creates one class of traffic. To create a flow of data we must specify a source/destination pair, the starting and ending times for this flow, and the maximum amount of bytes to be sent.

Castadiva also includes routing agents for well-known routing protocols, such as the AODV [6] provided by the Uppsala University and the OLSR [7] which is included with the OpenWRT distribution.

3.2 Main application

Castadiva's core element, a Java application running at the server, includes all the control functions required for test-bed experimentation. It is responsible for network topology maintenance, traffic control, as well as reporting the obtained simulation results and graphical representation. When the user defines the network scenario and the selected traffic, Castadiva configures 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 experiments are finished, Castadiva's core calculates the performance results, and finally it shows 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. In its development we have used the Model - View - Controller [8] design as reference.

3.2.1 Scenario Generation Tool

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. We now offer more details about Castadiva's GUI.

Main menu

A standard menu allows accessing the different options of Castadiva. Basic options were added, allowing a user to save and load a project, or export it to other test environments such as ns-2. It actually generates all the files required as input to this particular simulator, allowing to compare Castadiva's test results with those obtained through simulation. By selecting the Application menu you may start a new testbed experiment, load a previous one, or save the last one defined. It also allows to exchange files between ns-2 and Castadiva, allowing to use the same scenario and traffic in both applications.

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. The MAC address of each node 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. 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 color.

We may edit node properties, such as position and signal range, and also we can activate or deactivate the RTS/CTS 802.11 option for each node. Castadiva offers scenario option editing, where we can define the scenario bounds, the test time, node mobility and the routing protocol used. The Declare traffic button allows setting traffic, and the stop button halts it. Finally, a status bar provides general information to inform the user about what is being done.

Mobility in Castadiva. It is important to point out the speed and pause option of the Scenario Window. If a user write a value greater than zero in the speed option, each node acquire a random movement with a speed between zero and the inserted value. When a node arrives to a destination point, it wait for a selected pause time and then select a new random destination point to move on. This behavior is also completely compatible with the one selected in the ns-2 simulator.

Castadiva generate all node movements needed for the simulation before it starts. For each simulated second, it calculates the visibility range for each node. The obtained visibility is translated into iptables rules and sleep commands. Algorithm 1 shows the behaviour of Castadiva when a node (with MAC 00:14:BF:3C:39:EA) go out range at second 10 and return in range at second 25.

Castadiva also allows a user to capture the network topology at any simulated second, which could be useful to do a post-processing of the changes occurred in a network topology

\begin{algorithm}
\caption{Iptables rules for a simulation between two nodes.}
\begin{verbatim}
sleep 10
iptables -I INPUT -m mac --mac-source 00:14:BF:3C:39:EA -j DROP
sleep 15
iptables -D INPUT -m mac --mac-source 00:14:BF:3C:39:EA -j DROP
\end{verbatim}
\end{algorithm}

when the mobility has been activated.

3.2.2 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. When an experiment finishes, Castadiva fills in this table with results, including throughput, and, if traffic is UDP based, the percentage of packets correctly received.

3.2.3 Random test generator

Sometimes it is useful to automate the test-bed 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.

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 allows also 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 modi-
4 Performance Evaluation and Validation of Castadiva

To verify that the proposed tool behaves correctly, and to test its functionality, we have chosen a representative scenario where nodes are located so that the maximum number of hops between nodes is of four. The scenario is defined in a 1000m x 700m area, and the test time is of 510 seconds. We set the wireless nodes’ range to 250 meters. In terms of traffic, we define both UDP and TCP connections between each participating node and node AP7. For TCP connections, the maximum transfer size is of 1000 MB. UDP flows generate 4 packets per seconds, and packet size is fixed at 512 bytes. The traffic start at the simulation time of 30 seconds, allowing some previous node movement.

Figure 4 shows a first comparison between Castadiva and ns-2 node by node. In this test, each node has a maximum speed of 5 m/s, and no routing protocol is selected, therefore only traffic between directly connected nodes is allowed. The selected scenario was generated by ns-2 and imported to Castadiva.

The results show that the obtained results are quite similar, which validate the Castadiva implementation. Since we have no selected any routing protocol, only those connections which go through directly connected nodes can successfully delivered. We also observe that Castadiva has a lower throughput than ns-2. When Castadiva is selected, the shared wireless media is prone to both transmission errors and contentions among stations. In the case of ns-2, only contention effects are simulated, which explains the observed discrepancy.

Now we evaluated the impact that node speed has over the aggregated UDP and TCP traffic. We vary node speed among 0, 5, 10, 15 and 20 m/s and repeated all the test both in Castadiva and using ns-2. Figure 5 shows the obtained results. We observe that when we selected TCP traffic the differences among Castadiva and ns-2 increase. In TCP, when a packet is lost or it arrives out of order, it is transmitted again. We observe that the initial retransmission timeout differs between Castadiva and ns-2 and so the obtained differences. In ns-2 the timeout is around 5 second while the TCP implementation of Castadiva uses a timeout of 8 simulated seconds. We now repeat all the previous experiments but enabling the OLSR routing protocol, so that nodes behaves just as routers. We configure the OLSR parameters according to those proposed in the RFC standard. We set the HELLO_INTERVAL, the REFRESH_INTERVAL, the TC_INTERVAL, the MID_INTERVAL, and the HNA_INTERVAL according to those presented in Table 1. The obtained results shows that the average percentage of UDP packets correctly received increases since now all the selected connections can go ahead.
using a multihop communication. When we select TCP traffic, we observe that the average throughput does not increase, because the network was already saturated. However, since we are using the OLSR protocol, now the throughput is shared among all the nodes. We also observe that when UDP traffic is selected, as node speed increases, the packet delivery ratio decreases both in ns-2 and in Castadiva. When the traffic is based on TCP flows, the obtained results are mainly affected by the bandwidth and the behavior of TCP.

![UDP throughput comparison with NS-2 testing different node speeds](image1)

![TCP throughput comparison with NS-2 testing different node speeds](image2)

Figure 5: Impact that node speed has over the aggregated traffic in Castadiva and ns-2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value used</th>
</tr>
</thead>
<tbody>
<tr>
<td>HELLO_INTERVAL</td>
<td>2</td>
</tr>
<tr>
<td>REFRESH_INTERVAL</td>
<td>2</td>
</tr>
<tr>
<td>TC_INTERVAL</td>
<td>5</td>
</tr>
<tr>
<td>MID_INTERVAL</td>
<td>TC_INTERVAL</td>
</tr>
<tr>
<td>HNA_INTERVAL</td>
<td>TC_INTERVAL</td>
</tr>
</tbody>
</table>

Table 1: OLSR Parameters.

5 Conclusions

In this paper we present Castadiva, a novel architecture to improve research in the MANETs field by allowing to make real test-bed experiments in a simple and straightforward manner.

Using both TCP and UDP data traffic and under a wide variety of static and dynamic MANETs scenarios we compare Castadiva with the ns-2 simulation tool and show that our Castadiva test-bed is able to obtain confident results while using real wireless off-the-shelf devices.

References


