CupCarbon-Lab: An IoT Emulator

Ahcène Bounceur[1], Olivier Marc[2], Massinissa Lounis[1], Julien Soler[2], Laurent Clavier[3], Pierre Combeau[4]
Rodolphe Vauzelle[4], Loic Lagadec[5], Reinhardt Euler[1], Madani Bezoui[1,6] and Pietro Manzoni[7]
[1]Lab-STICC CNRS UMR 6285 - Université de Bretagne Occidentale, Brest, France
[2]Virtualys, Brest, France
[3]IEMN, IRCICA USR 3380, Lille, France
[4]Xlim CNRS UMR 7252 - Université de Poitiers, France
[5]ENSTA, Brest, France
[6]University of Boumerdes, Algeria
[7]Universitat Politècnica de València, Spain

***Abstract*—In the new generation of networks, not only computers are connected but also objects like cars, streets, buildings, or just, everything. In general, these things communicate by means of internet using gateways. Simulating these systems can help to validate specific algorithms and concepts. However, it cannot give any accurate information about reality when some problems arise such as delays, disconnectivity, attacks, insecurity, and especially in the case of large networks. Also, the real implementation of such networks is very complex, time consuming and can be impossible when the nodes are situated in different cities or countries. In this demo we propose a new platform called CupCarbon-Lab based on the existing simulator CupCarbon, where the codes used in simulation can be directly injected in real connected embedded cards like Raspberry Pi cards. This platform can automatically generate from the software a real IoT network even it is already deployed, which can be reconfigured without the need to go through each node. It also helps to test the feasibility and the scalability of an algorithm in real conditions.**

***Index Terms*—IoT, Simulation, Emulation, Network generation.**

I. Introduction

The IoT (Internet of Things) appeared with the new generation of internet that allows to mobiles and communicating devices to be connected via the web in the same way as computers. It is now possible to remotely control or monitor any object such as a robot, blood glucose meter, homes, cars, etc. Thus, an unlimited number of applications can be created to improve our daily life and especially that of people with reduced mobility.

Deploying several IoT nodes is very difficult, costly, time consuming, and can be impossible when nodes have to be reprogrammed frequently or when nodes are deployed in distant locations, for example in different cities or countries. Thus, the use of software to generate, reconfigure, and monitor such networks is very practical and useful. A first platform called CupCarbon-Lab is developed for this purpose. In this demonstration we will present this platform which is based on the CupCarbon simulator environment[1] and which can help to design and implement real networks efficiently with less effort, time and money. The IoT nodes, based on Raspberry Pi cards, can be installed and deployed physically in their real place after which it is possible to configure and program each node from the CupCarbon platform.

[1]`http://www.cupcarbon.com`

II. CupCarbon Simulator

A. SenScript

SenScript is the script used to program sensor nodes of the CupCarbon simulator. It is intuitive and close to the natural distributed programming language. The nodes can be programmed also with Python for more flexibility and options. The following script shows an example of a SenScript code.

```
1: atget id cid
2: loop
3:    wait
4:    read x
5:    if($x < $cid)
6:       mark 1
7:    else
8:       mark 0
9:    end
```

The command (`atget id`) of line 1 allows to assign to the variable cid the identifier of the current node. The command (`loop`) allows to start the loop section, where all the code situated after will be executed an infinite number of times. The command (`wait`) will allow to wait for a received message. This command is blocking and the next code will not be executed until a message is received. The command (`read`) of line 4 will assign the message received in the buffer to x. In line 5, we test if the received message (an identifier) is less than the value of the current identifier cid. If this is the case, line 6 will be executed, and the node will be marked (`mark 1`), otherwise, line 8 will be executed, where the current sensor will be unmarked (`mark 0`).

B. CupCarbon

An existing version of the simulator CupCarbon is available online. Figure 1 shows its main graphical user interface. It is ergonomic and designed to be used easily and intuitively.

The 2D and 3D visualization is very important for the deployment of the different nodes of the network. The 3D environment helps to do an accurate deployment where the elevation can be taken into account. This elevation generates

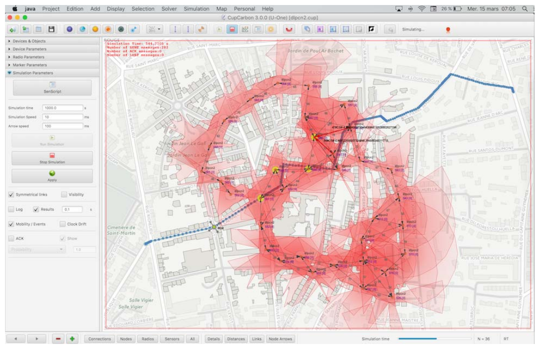

Fig. 1. Main parts of the architecture of the CupCarbon platform.

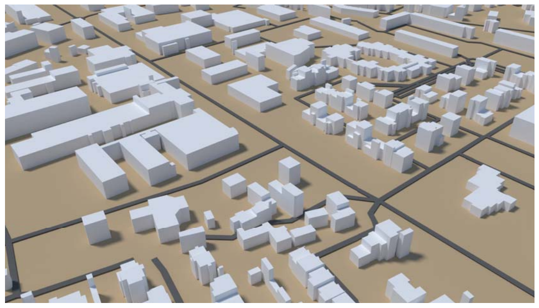

Fig. 2. Example of a 3D part of a city as displayed in CupCarbon.

different radio propagation and interferences. The 2D environment is useful for simulation debugging and validation. Figure 2 shows an example of a city displayed in the 3D environment of the simulator CupCarbon.

The 3D environment of CupCarbon is composed of ground elevation, buildings and various objects like sensor nodes. The ground elevation can also be obtained using external web services such as Google Elevation API. Since CupCarbon is programmed in Java, we have used the JOGL (*Java OpenGL*) library in order to create and use the OpenGL context.

III. CupCarbon-Lab

CupCarbon-Lap represents an extension of the simulator CupCarbon. As shown by the architecture of Figure 3, a server is added in order to generate a network by injecting the codes into real nodes (Raspberry Pi, Android, etc.) and to manage the communications between these nodes. Also, a node can be virtual and can communicate with the real sensors, which allows then to run hybrid emulation.

The platform integrates interpreters of the SenScript and the Python codes which are injected directly in real hardware platforms. The Python is integrated in order to be able to inject it in other kind of cards, like LoPy cards. As shown by Figure 4, each instruction is executed locally except the communication instructions like the transmission and the reception which are executed by the server. In the case of wireless communications, the messages are received directly by the nodes without going through the server. The server will then be informed in order to display the information on the screen.

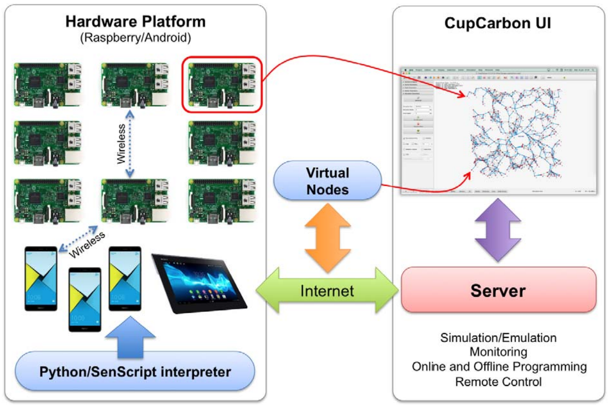

Fig. 3. Architecture of the CupCarbon-Lab.

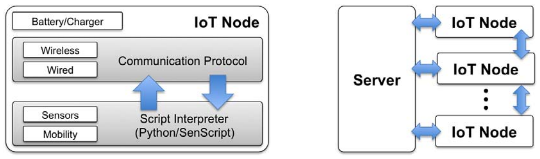

Fig. 4. The IoT node.

This demonstration is based on a simple architecture where a server is managing all the integrated nodes. It will serve as a proof of concept that allows to demonstrate the feasibility of the proposition. Other architectures can be developed in the future, where we will add the possibility of a direct, ad hoc, communication between nodes using the WiFi protocol. Figure 5 shows an example of the real platform where each node is composed of a Raspberry Pi card, a battery and a sensor platform. A node can also be an Android Smartphone/Tablet, a robot, etc. With this possibility, it is then possible to test the limitations of the remote controlling or the mobility in real conditions using the proposed platform.[2]

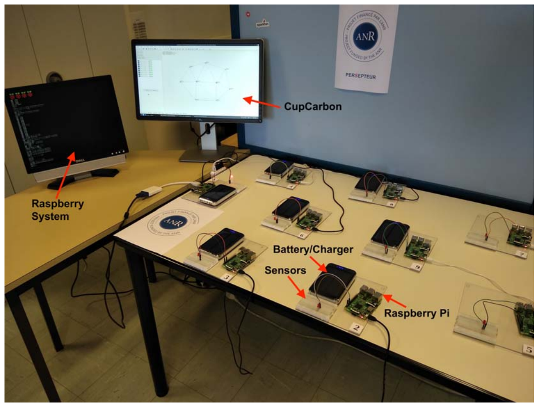

Fig. 5. The real platform.

[2]This project is supported by the French National Research Agency ANR PERSEPTEUR - REF: ANR-14-CE24-0017.