Project RobAIR2013, Group 1
WiFi roaming

Laurène Guelorget and Alexandre Corso

April 11, 2013

Introduction

RobAIR is a project in which several schools and departments are involved, such as the RICM and 3I departments of Polytech'Grenoble, ENSIMAG and Pole Design de Villefontaine. This project consists of developing an extensible and open-source robotic platform. The robot, called RobAIR, is a telepresence robot that can be used for museum tours while the user controls it thanks to an Android tablet. The conception of this robot was divided into several sub-projects, such as the electronic aspects for the 3I students, WiFi Roaming for the RICM's group 1 students, the robotic aspects for the RICM's group 2 students and the communication between the user and the robot for the RICM's group 3 students. As the members of the RICM's group 1, we will explain our sub-project in this report.

1 WiFi roaming

1.1 What is WiFi?

WiFi is a wireless network technology that allows an electronic device to exchange data wirelessly over a computer network. WiFi is a local area network (LAN) that uses high frequency radio signals to transmit and receive data. WiFi allows us to connect computers, phones or other high-speed throughput devices over distances of a few dozen meters (between 20 and 50 meters). For WiFi to function, several information are necessary, such as the Service Set Identifier (SSID), which basically is the name of the WiFi network, the channel used, and possibly a password.

WiFi permits the establishment of connections between devices by two modes:

- With the ad-hoc mode, equipments communicate directly, without intermediate.
- With the infrastructural mode, computers are connected thanks to an Access Point. The Access Point is an obligatory central point to all the communications.

For our project, the infrastructural mode is used because RobAIR will be used in a museum where every room is likely to have its own WiFi Access Point.

1.2 What is Roaming?

Roaming is a technology which allows a station to change his Access Point while remaining connected to the network. Roaming was not implemented in the WiFi standard, even though WiFi is a wireless technology. The reason is that WiFi exists since 1999 but at the time, wireless stations (such as laptops) were moved only when turned off. So contrarily to the GSM network, which included roaming from the start, roaming is not integrated to the WiFi technology.

This technology is a very important point concerning our project as we have to ensure that the robot stays connected at all time. As we already pointed out in this document, RobAIR will be used in a museum where every room is likely to have its own WiFi Access Point. When the robot changes room, it has to stay connected, even as the change of Access Point is being made.

In the figure 1, we can see how roaming will work in our project. Our robot possesses two interfaces (`wlan0` and `wlan1`) and at all time, it is connected to at least one Access Point. First, the robot is connected to the *Router1* on `wlan0`. By moving, the robot receives signals from other Access Points. The *Router2*'s signal is stronger so the robot connects its free interface (in this example, `wlan1`) to *Router2*. It can then disconnect itself from the *Router1* Access Point. This process goes on as the robot moves around.

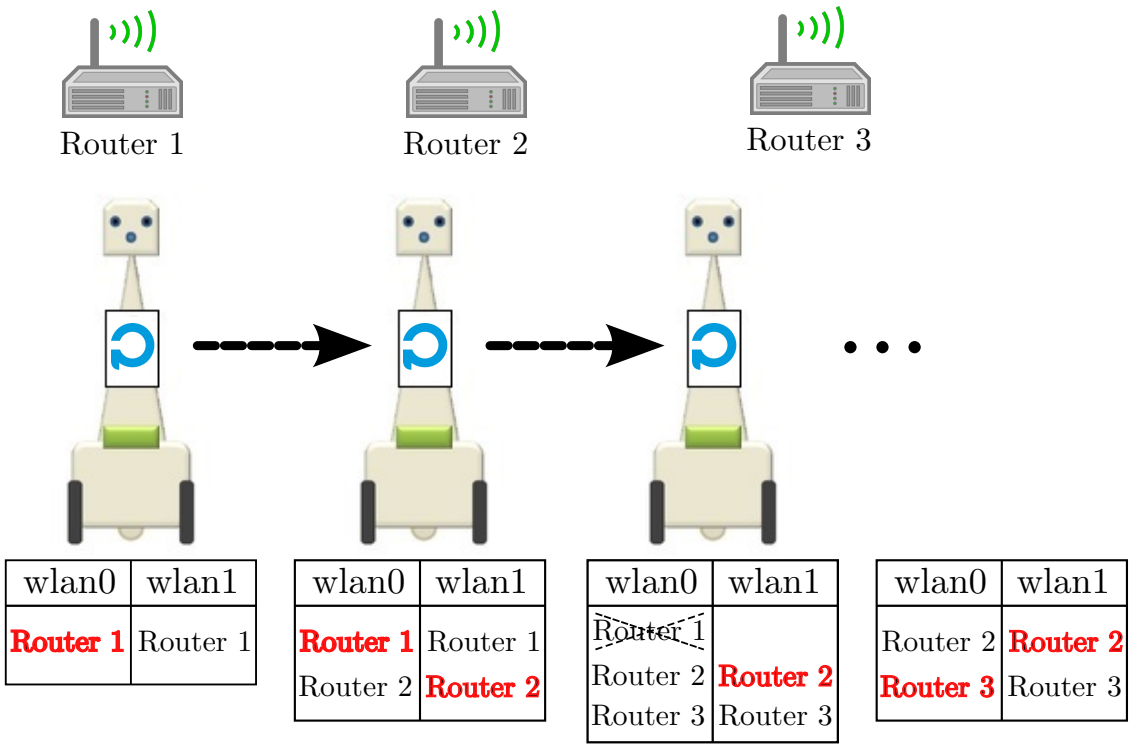

Figure 1: roaming process

2 Our program

To ensure that the robot stays connected at all time, we wrote a program in language C. We created several data structures to manage the WiFi networks that we will explain later in this report. This program of WiFi roaming will be executed on the *Ubuntu* tablet that will be on the robot.

2.1 General principles

Our program follows several steps:

- First, the program scans and searches for all available networks in range thanks to the `iwlist` command.
- After that, the program puts the SSID, the quality of the signal (the signal strength) and the index (an ID number to identify the WiFi network) in a file.
- Then, by parsing this file, the list of available networks are ordered by signal strength into a data structure (cf. section 2.2.2)

- Next, we compare the SSID to a database of past used and known networks to get the password for the WiFi network.
- Finally, we connect the robot to the network.

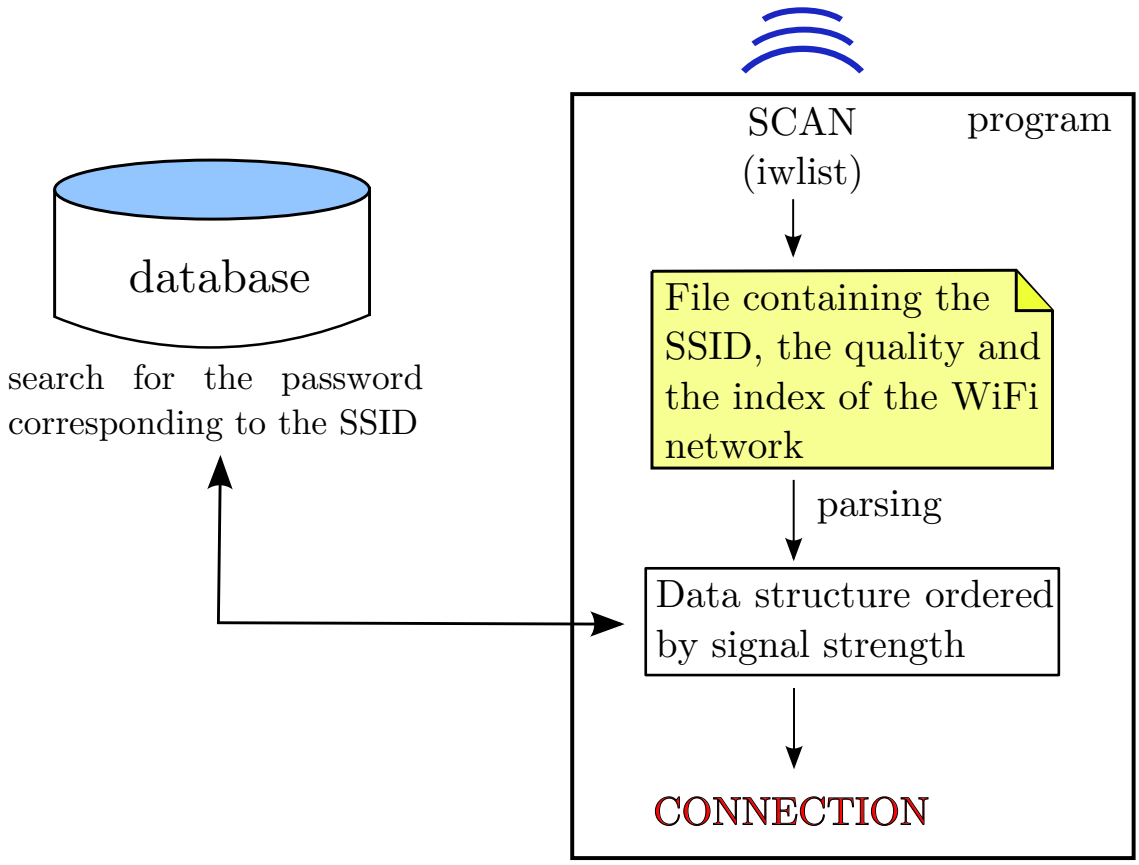

Figure 2: global functioning of our program

This process is implemented into a loop and is repeated at regular time intervals. If a WiFi network scanned is not in the database, the process restart with a reduced time lapse. If one interface is already connected to a WiFi network, then we try to connect the other interface. In any other case, we connect the interfaces in turn, one after the other. We save the SSID of the network on which an interface is connected in a variable so that we can't connect an interface to a WiFi network already used by the other interface.

2.2 Implementation

2.2.1 UNIX command

Our program uses a UNIX command, `iwlist`. The command `iwlist` allows us to view lots of information about a wireless card and its connectivity. We use the scan option which is a privileged operation (*root* only). This option gives us the list of Access Points and Ad-Hoc cells in range, and information about them (such as the SSID, quality, frequency, type of encoding, etc.).

example of use: `iwlist wlan0 scan`

2.2.2 Data structures

We created four data structures, one of them a linked list.

```
typedef struct{
   char *interface1;  //the first interface
   char *interface2;  //the second interface
   wifiPoint_t *lastWifiPointInt1;  //interface1's WiFi
   wifiPoint_t *lastWifiPointInt2;  //interface2's WiFi
   char *nextInterface  //next interface to be connected
}infoWifi_t;
```

First, we created the `infoWifi_t` structure. This data structure contains the interfaces of the robot, the Access Point on which each interface is connected (equals to `NULL` if an interface is not connected to any WiFi network) and the next interface to be connected to a WiFi network.

```
typedef struct{
    int index;  //id of the WiFi network
    char ssid[100]; //SSID of the WiFi network
    int quality;  //quality of the signal = signal strength
} wifiPoint_t;
```

The `wifiPoint_t` structure contains the SSID, the quality of the signal (which equals to the signal strength) and the index of a WiFi network (which is a id number that identifies the network).

```c
typedef struct {
    wifiPoint_t *wifiPoint; //a WiFi network
    char password[100]; //password of this AP
} wifiPointPwd_t;
```

Then, we created the `wifiPointPwd_t` structure. This data structure contains a WiFi network and its password.

```c
typedef struct _listWifi_t{
    wifiPoint_t *wifiPoint; //a WiFi network
    struct _listWifi_t *previous; //last AP in the list
    struct _listWifi_t *next; //next AP in the list
} listWifi_t;
```

Then, we created the `listWifi_t` structure. This data structure is a linked list of the WiFi networks.

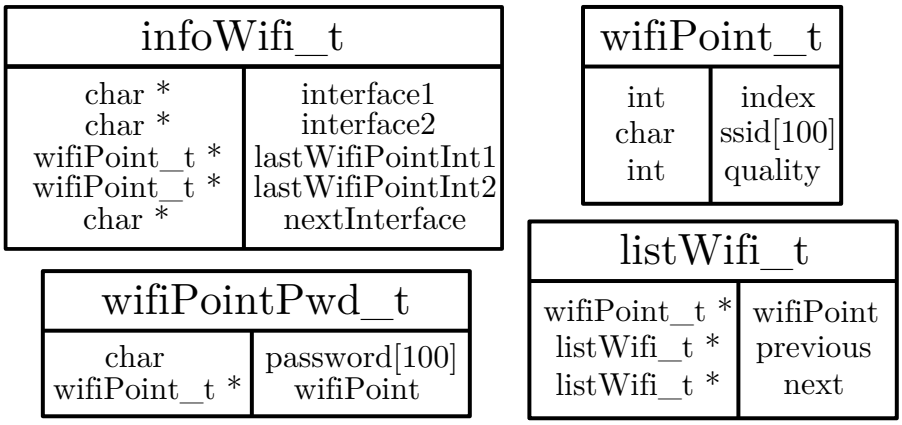

Figure 3: data structures for our program

3 Multipath Transmission Control Protocol

We decided to use the Multipath Transmission Control Protocol (shortened MPTCP in the rest of this report) because it offers better throughput and a smoother reaction to failures.

3.1 Generalities

MPTCP adds the capability of simultaneously using multiple paths to a regular TCP session. The availability of multiple connection paths and the capability to dynamically schedule traffic between them allows more flexibility and efficiency in Internet congestion control.

MPTCP is a standard in the pipeline at the Internet Engineering Task Force (IETF). The goal of this protocol is to use the same TCP connection through different network interfaces. One of the most typical use is to unload GSM networks via WiFi. It allows you to use WiFi on your Smartphone and as soon as you are not in range of WiFi, you swap to GSM 3G network. The main advantage of MPTCP is that the change of network is totally transparent for the applications.

Even if the standard is in the pipeline, it is already possible to use MPTCP on a Linux distribution thanks to a modified version of the kernel. Hence, we had to recompile the kernel to be able to use MPTCP. This is the reason it is not yet possible to use MPTCP in the "TP de réseaux", recompiling the kernel taking too much time.

3.2 Functioning

First, there is a handshake with the server, like with the classic TCP protocol. Only with MPTCP, the client sends a additional option that signals that it is capable to use MPTCP. The server has to respond with the same option for the communication to be enabled. The congestion management with MPTCP aims to have the throughput at least as good as the max throughput on the best interface. For that, a congestion window is kept for each path. After that, the congestion management is the same as in classic TCP.

To make sure MPTCP was better that TCP, we conducted our own tests. For that, we used the following configuration:

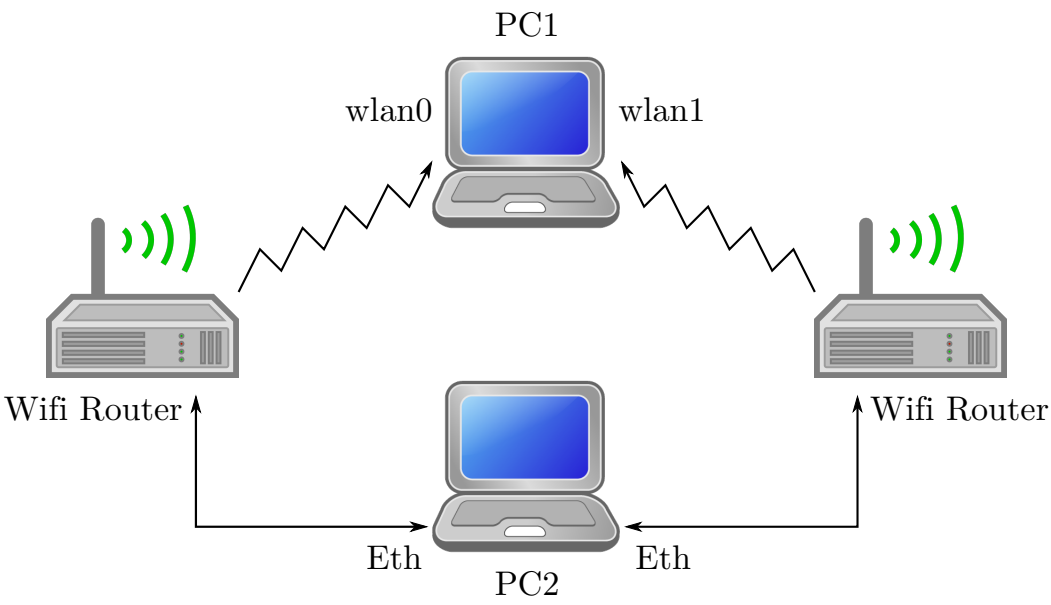

Figure 4: configuration used for our tests

We compared the latency when disabling an interface on PC1 with TCP and MPTCP. The results showed that the latency while using MPTCP is lower than while using classic TCP.

Conclusion

To conclude, we wrote a program capable to scan the networks in range and obtain information regarding them, such as the SSID, the type of encoding, the signal strength, etc. So far, our program supports networks with WEP and WPA encoding. This project allowed the three groups to develop individual tools that eventually could be assembled together. This establishes a ground base for students in RICM3 that will work on the RobAIR2014 project. They will be able to understand much more quickly the main principles of ROS, thanks to the tutorials of the Group 2, and they will have a WiFi roaming program available to use.