



UNIVERSITAT<sup>DE</sup>  
BARCELONA

---

## Pràctica 4: Comunicacions a través de la pila TCP/IP

---

Noah Márquez & Jan Morales

17 desembre 2023

# ÍNDICE

<b>1</b>	<b>Introducció</b>	<b>3</b>
1	Objectius . . . . .	3
<b>2</b>	<b>Treball realitzat al laboratori</b>	<b>4</b>
1	Creació del canal recol·lector de dades . . . . .	4
1.1	Firmware de la mota . . . . .	5
1.2	Explicació detallada del codi . . . . .	5
2	Creació d'un punt d'accés . . . . .	8
3	Client-Servidor . . . . .	8
3.1	Servidor . . . . .	9
3.2	Client . . . . .	10
4	Simulació amb Packet Tracer . . . . .	11
4.1	Topologia de xarxa . . . . .	15
4.2	<i>Sniffer</i> i protocols de xarxa . . . . .	16
<b>3</b>	<b>Conclusions</b>	<b>17</b>

## 1. INTRODUCCIÓ

El model TCP/IP és una especificació de protocols de xarxa creat per Vinton Cerf i Robert E. Kahn, a la dècada dels 70. Va ser implantat a la xarxa ARPANET, la primera xarxa d'àrea àmplia (WAN), desenvolupada per encàrrec de DARPA (Departament de Defensa dels Estats Units) i predecessora d'Internet; per aquesta raó, de vegades també se'n diu model DoD o model DARPA.

Aquest model resulta de molta importància alhora d'explicar com funciona una xarxa des del nivell més baix. En aquesta pràctica es tractarà sobre aquest model, aplicant els coneixements adquirits sobre aquest per realitzar una part pràctica amb ajuda de hardware i software proporcionat pel professorat.

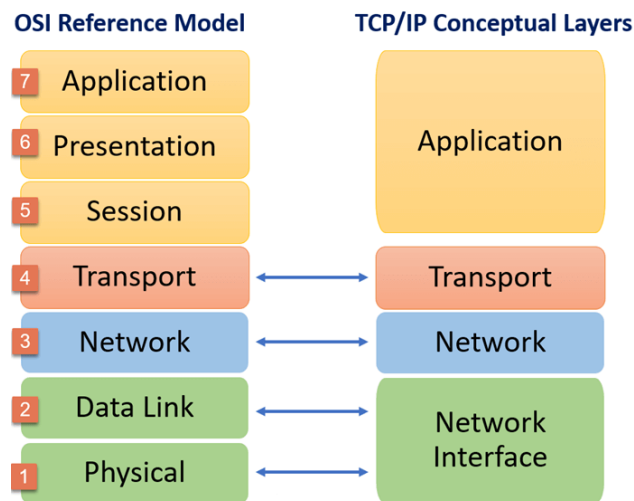


Figura 1.1: Relació model OSI i TCP/IP

### 1. Objectius

L'objectiu de la pràctica és conèixer el funcionament del protocol TCP/IP i treballar sobre aquesta pila per muntar un sistema de transferència de dades entre un dispositiu i un servidor, analitzant com són els paquets necessaris.

Per tal d'assolir l'objectiu, treballarem amb una doble vessant. Per una banda la primera part de la pràctica estarà basada en els dispositius que hem fet servir a la pràctica anterior. Aquesta primera part de la pràctica es realitzarà amb l'ajut dels professors de pràctiques de l'assignatura.

Per altra banda, per tal d'analitzar el comportament dels diferents protocols ens instal·larem un software de simulació que ens permetrà l'estructura dels diferents paquets que estem enviant per la xarxa. El simulador que farem servir és el Packet Tracer de Cisco. Aquesta part de la pràctica serà prèviament explicada pels professors de l'assignatura.

## 2. TREBALL REALITZAT AL LABORATORI

### 1. Creació del canal recollidor de dades

En la primera part de la pràctica ens centrarem en enviar unes dades a partir del dispositiu hardware proporcionat pel professorat. Amb aquesta finalitat crearem un canal on agafar totes les dades que posteriorment seràn enviades des de la mota.

Per fer-ho crearem un canal per guardar dades de la nostra mota i poder veure les dades que estem recollint. Això ho farem des de **Thingspeak**, una nova eina que ens permetrà crear canals amb aquesta finalitat. El primer que necessitarem és crear un compte amb el correu de la Universitat i se'ns activarà una llicència educativa amb la qual podrem utilitzar els serveis que ofereix aquesta eina.

Un cop estem en disposició d'un compte crearem un canal, el nom del qual serà ESP8266\_Nom\_RSSI i l'informació que posem al camp 1 és RSSI, la resta de camps es deixaràn buits o amb els valors per defecte. Si el canal s'ha creat correctament es mostrarà una pantalla com la observada en la següent figura:

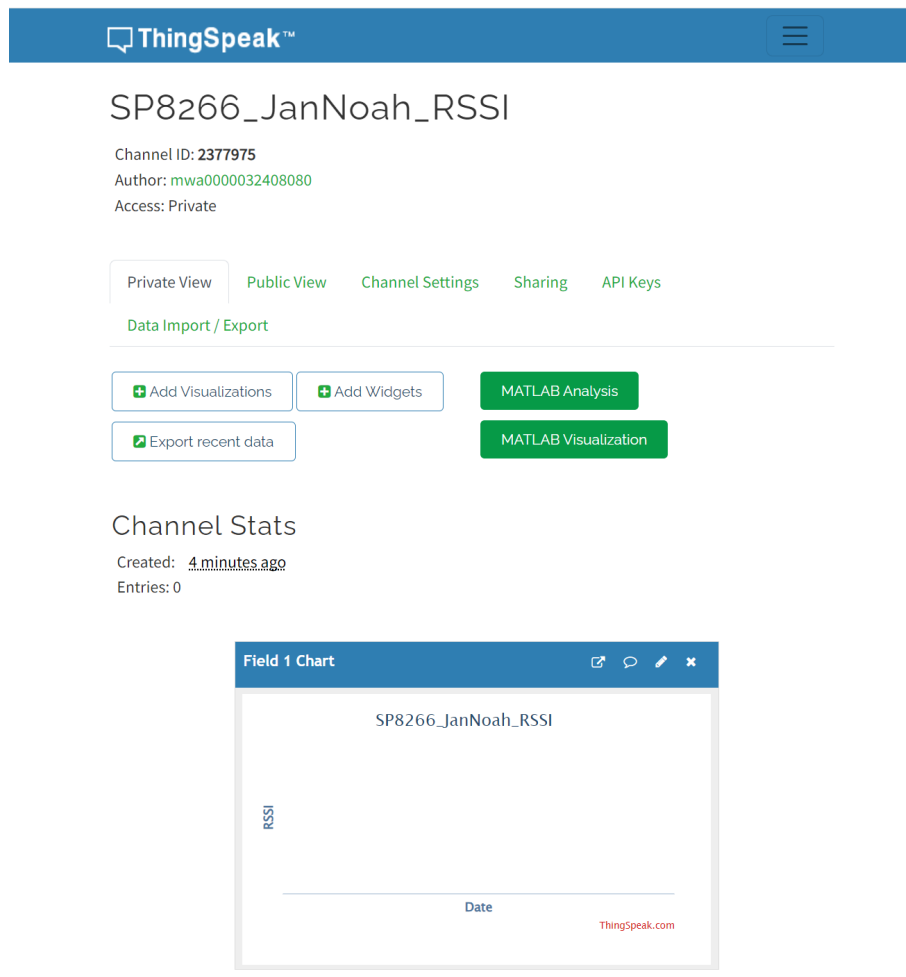


Figura 2.1: Canal de recollida de dades en *Thingspeak*

El següent pas és carregar a la mota un *firmware* específic amb l'objectiu de transmetre valors de RSSI obtinguts de forma similar a com ho feiem a la pràctica 2.

### 1.1. Firmware de la mota

Amb l'objectiu mencionat, dissenyarem un petit programa en Arduino on implementarem un menú molt senzill amb el qual poder escanejar les xarxes Wi-Fi disponibles, connectar-se a una i enviar a *Thingspeak* els valors RSSI obtinguts de la Wi-Fi a la que ens connectarem, amb possibilitat de realitzar la desconnexió en finalitzar l'enviament de les dades.

Inicialment declararem una variable del tipus *WifiClient*, la qual ens permet connectar-nos a *Thingspeak* a través d'una determinada IP i un port. A continuació, si es realitza correctament la connexió entre la mota i el servidor passem a construir el nostre paquet de dades. Les dades que transmetem són els valors RSSI del senyal Wi-Fi a la que estem connectats afegint-ho dins del paquet.

Per enviar el paquet de dades es necessari el *writeAPIKey* per així poder escriure les dades al canal de *Thingspeak* que prèviament hem creat. En cada iteració deixarem un temps de retard (20 segons) perquè no se superposin les dades a enviar i anar agafant les dades de manera continua.

Un cop executem el programa, recollim les dades i les enviem a *Thingspeak*, on el canal s'actualitza mostrant la gràfica que observem en la següent figura:

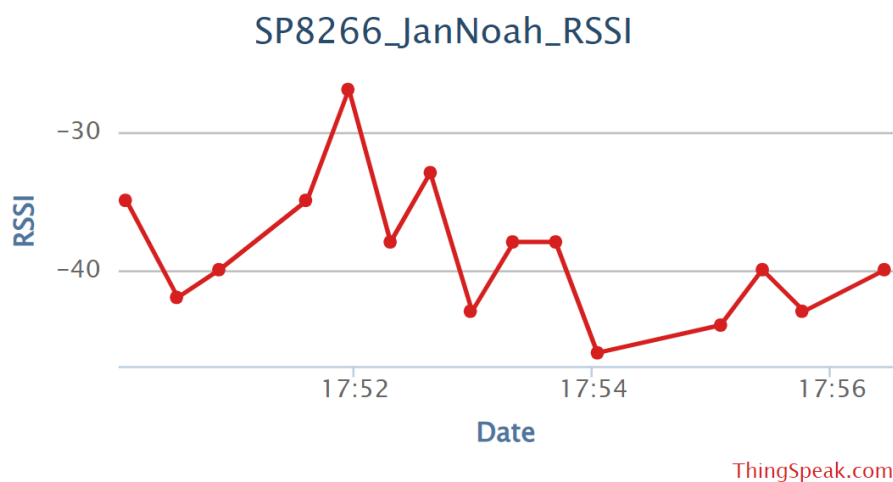


Figura 2.2: Canal de recollida de dades a *Thingspeak* actualitzat

### 1.2. Explicació detallada del codi

El següent codi d'Arduino està dissenyat per a ser utilitzat amb el mòdul *Wi-Fi ESP8266*. Aquest permet al dispositiu connectar-se a una xarxa Wi-Fi i enviar dades a *ThingSpeak*.

Per tal d'explicar detalladament el funcionament del codi, hem decidit afegir-hi comentaris:

#### Inclusió de la Biblioteca i Configuració Wi-Fi

```
#include <ESP8266WiFi.h> // Inclou la biblioteca necessaria  
                           per a la connectivitat Wi-Fi.  
  
// Definició de les constants per a la connexió Wi-Fi.
```

```
const char* ssid = "XXXX"; // El SSID de la xarxa Wi-Fi a la
                             qual ens connectarem.
const char* password = "XXXXXXXXX"; // La contrasenya de la
                             xarxa Wi-Fi.

WiFiClient client; // Crea una instancia de la classe
                   WiFiClient.
```

### Configuració de *ThingSpeak*

```
// Definició de les constants per a la connexió amb ThingSpeak
.
const int channelID = XXX; // ID del canal de ThingSpeak on s'
                             enviaran les dades.
String writeAPIKey = "XXXXXXXXXXXXXXXXXXXX"; // La clau API del
                             canal de ThingSpeak.
const char* server = "api.thingspeak.com"; // L'adreça del
                             servidor de ThingSpeak.
const int postingInterval = 20 * 1000; // Interval d'enviament
                             de dades en mil·lisegons.
```

### Funció *Loop*

```
void loop() {
    if (client.connect(server, 80)) { // Intenta connectar-se
        al servidor de ThingSpeak.
        long rssi = WiFi.RSSI(); // Obte la intensitat de la
                                    senyal de Wi-Fi.

        // Prepara el cos de la petició HTTP per a enviar a
        ThingSpeak.
        String body = "field1=";
        body += String(rssi); // Afegeix el valor de RSSI al
                                cos de la petició.

        // Mostra el valor de RSSI a la consola serial.
        Serial.print("RSSI: ");
        Serial.println(rssi);

        // Configura la petició POST HTTP per a enviar les
        dades a ThingSpeak.
        client.println("POST /update HTTP/1.1");
        client.println("Host: api.thingspeak.com");
        client.println("Connection: close");
        client.println("User-Agent: ESP8266");
        client.println("X-THINGSPEAKAPIKEY: " + writeAPIKey);
        client.println("Content-Type: application/x-www-form-
                        urlencoded");
        client.println("Content-Length: " + String(body.length
            ()));
    }
}
```

```

    client.println();
    client.print(body);

    client.stop(); // Tanca la connexio amb el servidor.
}

delay(postingInterval); // Espera un temps determinat
                        abans de realitzar la següent publicacio.
}

```

Al codi proporcionat en el enunciat de la pràctica hem hagut d'afegir el següent mètode *setup()* per tal de connectar-nos des de la mota a la nostra xarxa Wi-Fi:

```

void setup(){
  // Initialize serial communication for debugging
  Serial.begin(9600);
  // Connect to Wi-Fi
  WiFi.begin(ssid, password); // Coonecta el dispositiu a la
                               xarxa Wi-Fi utilitzant les credencials proporcionades.

  // Espera fins que la connexió Wi-Fi estigui establerta,
  // imprimint punts en la consola serial durant l'espera.
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected.");
}

```

El codi, per tant, es dedica a mesurar contínuament la força del senyal Wi-Fi i a enviar aquesta informació a un servidor extern per al seu seguiment i anàlisi, com podem observar a la següent:

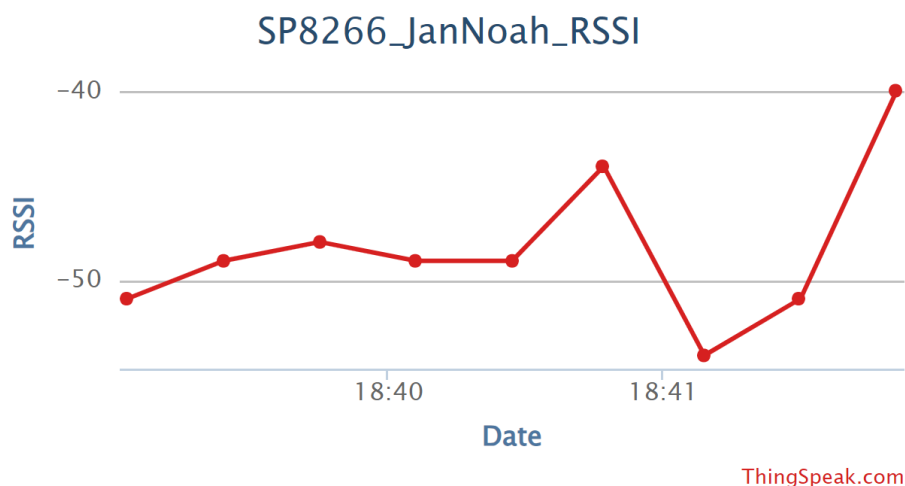


Figura 2.3: Canal de recollida de dades a *Thingspeak* actualitzat

Hem actualitzat el codi per utilitzar la biblioteca de *ThingSpeak* de la següent forma:

1. Hem inclòs la biblioteca de *ThingSpeak* al principi del programa per així utilitzar les fun-

cions específiques de la biblioteca *ThingSpeak*.

2. Al principi de la funció *setup()*, després d'establir la connexió Wi-Fi, hem afegit *ThingSpeak.begin(client)*; Això inicialitza la biblioteca *ThingSpeak* amb l'objecte client ja declarat.
3. A la funció *loop()*, hem substituït el codi que construeix i envia la petició HTTP manualment per una crida a la funció *ThingSpeak.writeField()*. Aquesta funció s'encarrega d'enviar les dades al nostre canal de *ThingSpeak*.

## 2. Creació d'un punt d'accés

El següent objectiu és crear un punt d'accés per tal de que sigui el nostre mòdul qui generi una xarxa Wi-Fi a la qual ens podem connectar i transmetre informació. Per crear un punt d'accés l'únic que haurem de fer es configurar la Wi-Fi que ens aporta la llibreria com un *Access Point*. Amb un altre mota podem veure que amb el codi de la practica anterior ens hem pogut connectar sense cap problema entre motes.

```
#include <ESP8266WiFi.h>

char ssid[] = "MyWiFi";
char password[] = "password";

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    Serial.println("WiFi access point test");
    WiFi.mode(WIFI_AP);
    WiFi.softAP(ssid, password);
    Serial.println("WiFi ON");
}

void loop() {
    // put your main code here, to run repeatedly:
}
```

Figura 2.4: Configuració del mòdul *ESP8266* com a punt d'accés

Fem servir la funció *softAP* per tal de passar el nom i la contrasenya de la Wi-Fi que volem crear. Amb això ja tindrem el nostre punt d'accés funcionant i ens podrem connectar a ell des de qualsevol dispositiu, per exemple altres motes.

## 3. Client-Servidor

A continuació construirem una arquitectura Client-Servidor per tal de comunicar dues motes on una actuï de Servidor i l'altra de Client, i així poder enviar diverses dades (com ara missatges). Per tant necessitarem crear dos codis, un per a cada mota, primer configurem el servidor i posteriorment el client que es connectarà a aquest servidor i enviarà dades.

Inicialment, en el nostre programa, configurem una placa en mode *WIFI\_STA(STATION)* per a poder connectar-la a l'altra placa que haurà d'estar configurada en *WIFI\_AP(Soft ACCESS POINT)* per a poder establir la seva pròpia xarxa Wi-Fi. D'aquesta manera estem inicialitzant una placa com a client i una altra com a servidor, respectivament.



### 3.1. Servidor

Primer configurem una de les dues motes com a servidor. Aquesta llavors, utilitzarà el mode de WIFI\_AP. La configuració de la placa és la que es mostra en la següent figura:

```
// Port del server
WiFiServer server(80);

void setup() {
  Serial.begin(9600);
  WiFi.mode(WIFI_AP); //Acces mode seleccionat
  WiFi.disconnect();
  WiFi.softAP(ssid,password);
  //WiFi.begin();
  delay(1000);

  server.begin();
  Serial.println("\nWiFi ON");
  Serial.print("Local IP: ");
  Serial.println(WiFi.softAPIP());
  Serial.print("Server port: ");
  Serial.println("80");
}
```

Figura 2.5: Configuració de la mota com a punt d'accés

A continuació inicialitzem la xarxa Wi-Fi amb la funció *WiFi.softAP(char \*ssid, char \*password)*. *ssid* i *password* fan referència al nom i a la clau de la xarxa, respectivament. Finalment ens disposem a obrir el servidor per tal de que estigui operatiu utilitzant la instrucció *server.begin()* de la classe *WiFiServer*.

L'objectiu de l'arquitectura que estem implementant és enviar un missatge al servidor i que aquest ho retorni a la resta de clients amb la finalitat de construir un xat entre les dues motes. La forma d'enviar el missatge es fent servir *POST Request* i un *JSON* com a *body* del paquet. Podem observar a la següent figura el fragment de codi que realitza la lectura de dades:

```
client.write(c);
if (c == '\n' && currentLineIsBlank) {

// Here is where the POST data is.
while(client.available())
{
    char c = client.read();
    data += c;
    client.write(c);
}
finish = true;
}
if (finish){
    finish = false;
    char c_data[data.length()];
    data.toCharArray(c_data,data.length()+1);

    deserializeJson(doc,c_data);
    const char *user_name = doc["UserName"];
    const char *message = doc["Message"];
    Serial.print(user_name);
    Serial.print(": ");
```

Figura 2.6: Fragment de codi que realitza la lectura de dades

### 3.2. Client

Com a segon pas, després de configurar el servidor, configurarem un client que només enviarà i rebrà missatges amb la finalitat d'imprimir-los per consola. Per enviar els missatges fem servir el mateix protocol que a l'apartat anterior utilitzant una estructura *JSON* per emmagatzemar la informació que volem transmetre. A la següent figura podem observar el mètode encarregat de transmetre el missatge:

```
client.write(c);
if (c == '\n' && currentLineIsBlank) {

// Here is where the POST data is.
while(client.available())
{
    char c = client.read();
    data += c;
    client.write(c);
}
finish = true;
}
if (finish){
    finish = false;
    char c_data[data.length()];
    data.toCharArray(c_data,data.length()+1);

    deserializeJson(doc,c_data);
    const char *user_name = doc["UserName"];
    const char *message = doc["Message"];
    Serial.print(user_name);
    Serial.print(": ");
```

Figura 2.7: Mètode per enviar el missatge

#### 4. Simulació amb Packet Tracer

L'última part de la pràctica té com a objectiu construir una petita xarxa entre diferents dispositius. Primer de tot assignarem una IP estàtica als nostres PCs i els connectarem amb dos routers connectats entre ells pel port sèrie amb un *switch* per a cada router.

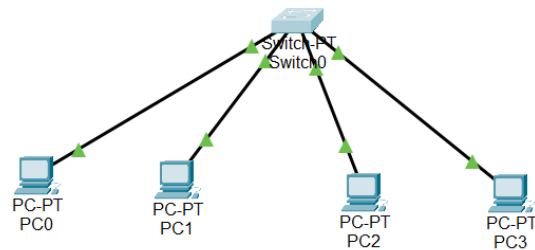


Figura 2.8: Esquema de les nostres connexions

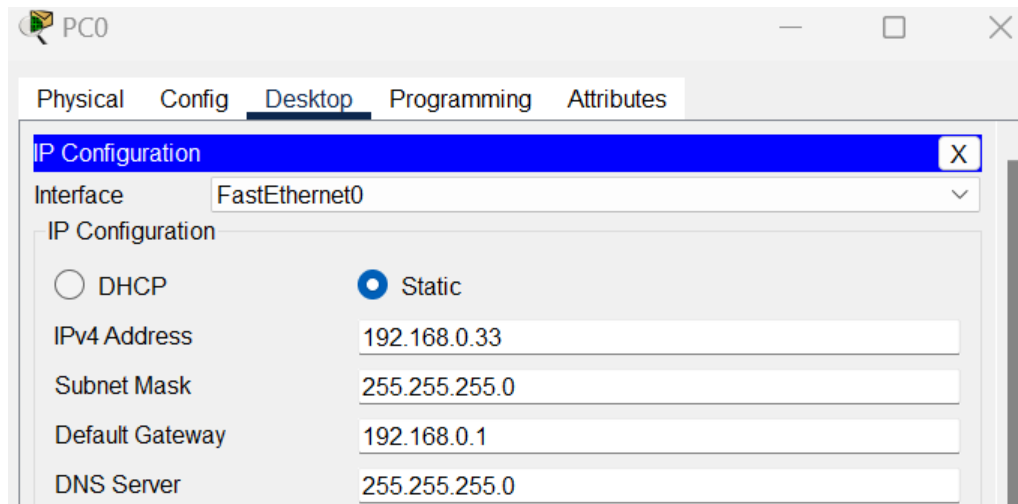


Figura 2.9: IP estàtica a cada un dels 4 PCs

A continuació comprovem que tot funciona fent un *ping* i mirem la simulació i la *Event List*.

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.34

Pinging 192.168.0.34 with 32 bytes of data:

Reply from 192.168.0.34: bytes=32 time=1ms TTL=128
Reply from 192.168.0.34: bytes=32 time<1ms TTL=128
Reply from 192.168.0.34: bytes=32 time<1ms TTL=128
Reply from 192.168.0.34: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.34:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Figura 2.10: PING al PC 1.

Com podem veure en la següent figura, que mostra la *Event List*, el que fa el PC des del qual fem *ping* és enviar un paquet a tots els PC. El PC que rep el paquet en qüestió respon i aleshores el primer fa connexió amb el segon.

Vis.	Time(sec)	Last Device	At Device	Type
	0.000	--	PC0	ICMP
	0.001	PC0	Switch0	ICMP
	0.002	Switch0	PC1	ICMP
	0.002	Switch0	PC3	ICMP
	0.003	PC1	Switch0	ICMP
	0.004	Switch0	PC0	ICMP
	0.515	--	Router7	CDP
	0.516	Router7	Sniffer0	CDP
	1.005	--	PC0	ICMP
	1.006	PC0	Switch0	ICMP
	1.007	Switch0	PC1	ICMP
	1.008	PC1	Switch0	ICMP
	1.009	Switch0	PC0	ICMP
	1.997	--	Switch0	STP
	1.998	Switch0	PC1	STP
	1.998	Switch0	PC0	STP
	1.998	Switch0	PC2	STP
	1.998	Switch0	PC3	STP
	2.009	--	PC0	ICMP
	2.010	PC0	Switch0	ICMP
	2.011	Switch0	PC1	ICMP
	2.012	PC1	Switch0	ICMP
	2.013	Switch0	PC0	ICMP

Figura 2.11: Event List

Ara procedirem a connectar la nostra xarxa amb un servidor a través d'un servidor web. Per a això seguim el tutorial pujat al campus virtual de l'assignatura que explica com configurar correctament aquesta connexió entre routers i núvol a la pestanya *Frame Relay*. Per tant primer hem de connectar els diferents serials per redirigir les IP que no coneguem cap a un altre xarxa.

Ara hauríem de dir-li al nostre *cloud* que identifiqui aquestes connexions i les connecti entre elles. Utilitzarem el *Frame Relay* per connectar-los entre si, a part dels serials del nostre router haurem d'activar l'encapsulació per *Frame Relay*. Per activar aquesta encapsulació introduïm a cada router la comanda mostrada en el tutorial del campus virtual:

```
Router(config-if)#encapsulation frame-relay
```

En la nostra xarxa local, les adreces IP de l'exterior són desconegudes i tot el que sabem és la informació d'un altre router. El que fem aquí és configurar el router per quan rebí una sol·licitud d'un ordinador o un servidor, poder localitzar fàcilment la informació que volem en un altre router i establir una connexió de comunicació.

Les dues imatges següents són les configuracions del router 0:

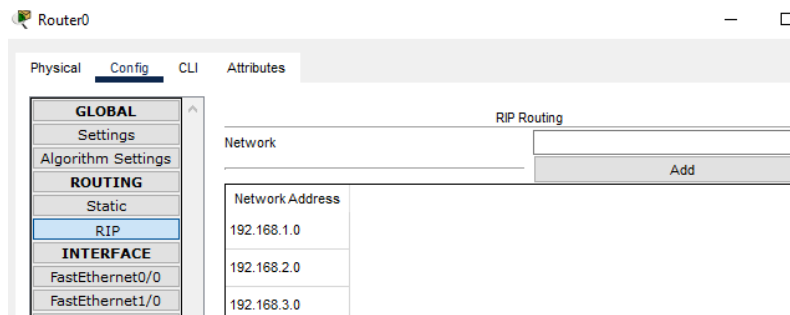


Figura 2.12: Router 0 RIP

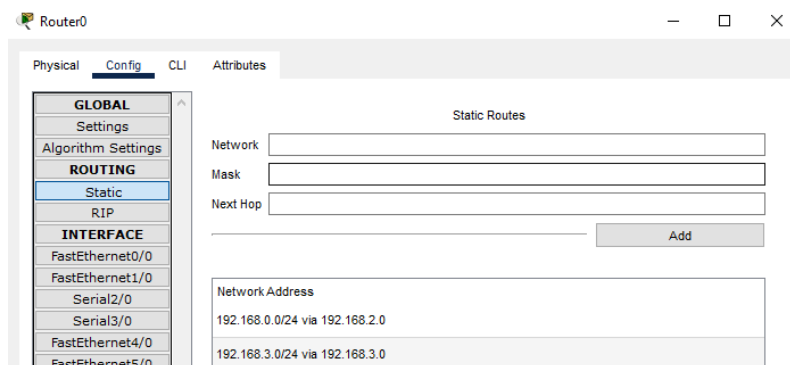


Figura 2.13: Router 0 static

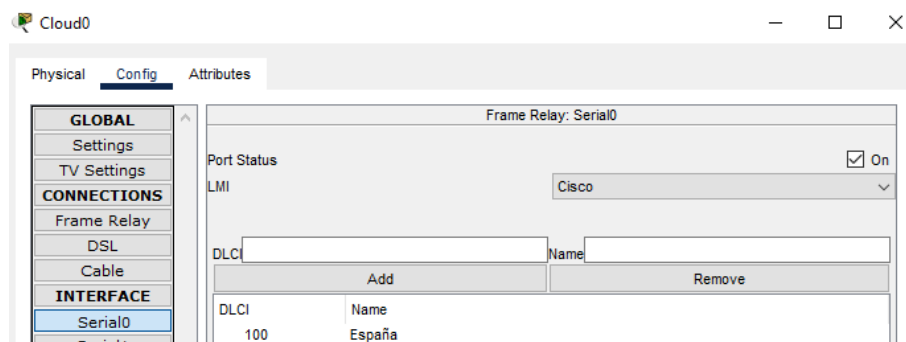


Figura 2.14: Serial 0

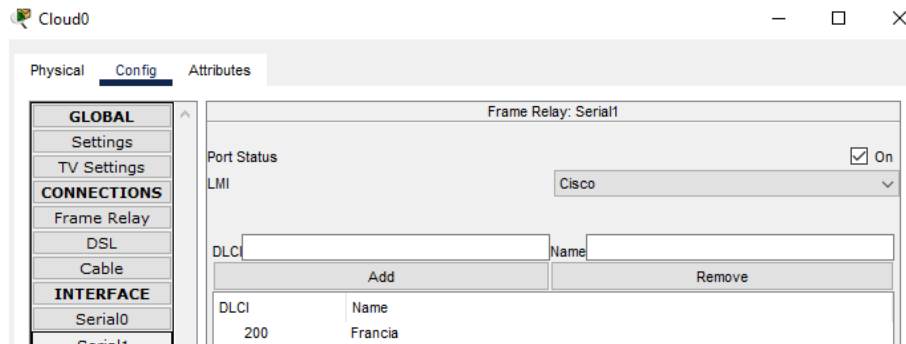


Figura 2.15: Serial 1

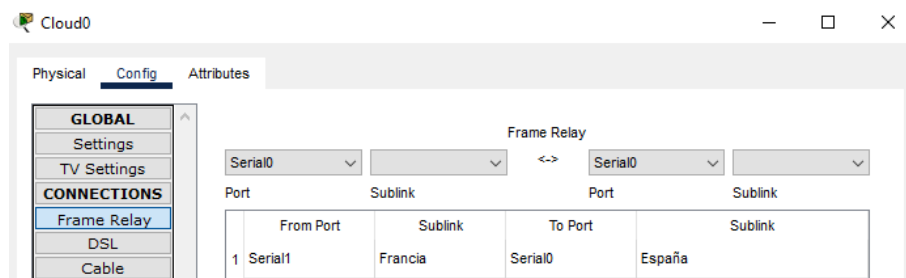


Figura 2.16: Frame Relay per connectar entre si

#### 4.1. Topologia de xarxa

Després de realitzar aquesta connexió configurem correctament el NAT tal i com indica l'enunciat de la pràctica. Finalment la topologia de la xarxa queda com s'observa en la següent figura. A part podem veure que en el primer router també hi posem un *sniffer*. Mitjançant l'*sniffer* podem comprovar l'estat de la informació.

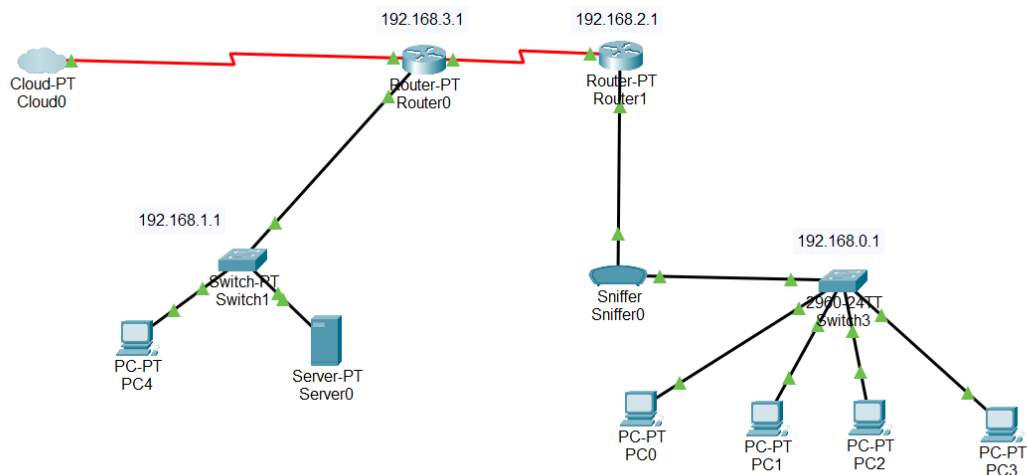


Figura 2.17: Topologia final de la WAN configurada

Podem comprovar que per exemple, hem enviat amb èxit un missatge del Servidor al PC:

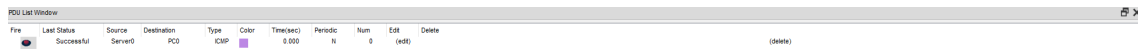


Figura 2.18: Server a PC

#### 4.2. Sniffer i protocols de xarxa

Mitjançant l'*sniffer* podem entendre l'encapsulació de la informació sota diferents protocols de xarxa informàtica. El primer protocol que utilitzem és el STP, el qual permet a les xarxes LAN i Ethernet tenir enllaços redundants en una LAN mentre soluciona els problemes coneguts quan s'hi afegeixen enllaços extres, i després fa un ICMP que és el protocol de Control de Missatges de Internet, apart el CDP és un protocol de Cisco que permet connectar dispositius entre si.

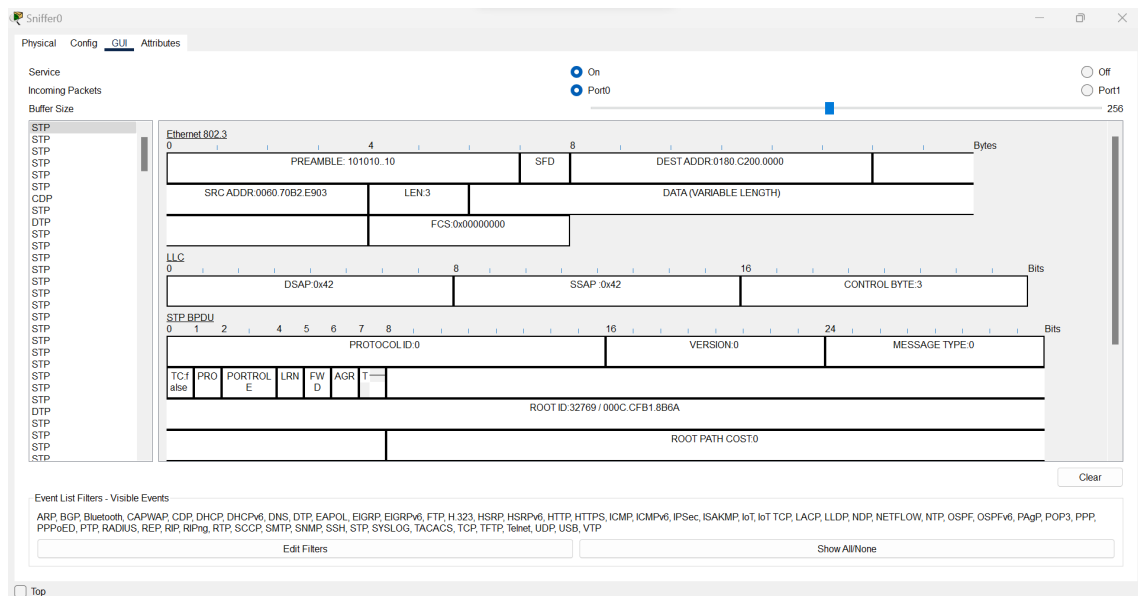


Figura 2.19: Sniffer i els seus protocols

En canvi en la primera part de la pràctica utilitzem protocols HTTP basats en TCP/IP.



### 3. CONCLUSIONS

Aquest projecte ha demostrat la capacitat de dissenyar i implementar un sistema de transferència de dades eficient utilitzant el protocol TCP/IP. A través de diverses fases, hem aconseguit els següents objectius:

1. **Construcció d'un Canal Recol·lector de Dades:** Mitjançant l'ús de l'eina *Thingspeak*, hem creat un canal per a la recollida i visualització de dades de RSSI des d'un dispositiu hardware. Això ha permès una recollida de dades eficient i la seva visualització en temps real.
2. **Desenvolupament de Firmware per a la Mota:** S'ha programat un *firmware* per a la mota ESP8266, que permet la recollida i l'enviament de dades de RSSI a *Thingspeak*. Aquest desenvolupament inclou la connexió Wi-Fi i la manipulació de paquets de dades.
3. **Creació d'un Punt d'Accés i Implementació Client-Servidor:** Hem establert un punt d'accés Wi-Fi i hem desenvolupat una arquitectura Client-Servidor per a la comunicació entre dispositius. Això ha permès la transferència de missatges i dades entre motes de manera eficaç.
4. **Simulació de Xarxa amb Packet Tracer:** Hem utilitzat *Cisco Packet Tracer* per simular una xarxa complexa, incloent la configuració de routers, assignació d'IP estàtiques, i l'ús de *Frame Relay* per a connectar xarxes. Això ha proporcionat una comprensió profunda dels aspectes de xarxat i l'encapsulació de paquets.
5. **Topologia de Xarxa i Anàlisi amb Sniffer:** La construcció d'una topologia de xarxa complexa, incloent l'ús de NAT i la implementació d'un *sniffer*, ha permès una anàlisi detallada del tràfic de xarxa i l'ús de diferents protocols com STP, ICMP, i CDP.
6. **Comparació entre Protocols:** Hem pogut comparar l'ús de protocols HTTP basats en TCP/IP amb altres protocols utilitzats en la xarxa, proporcionant una visió més àmplia sobre el comportament i l'eficàcia de diferents protocols en situacions variades.

En resum, aquest projecte ha aportat coneixements pràctics sobre el funcionament de TCP/IP i ha demostrat la importància de la planificació i implementació correcta en sistemes de transferència de dades. Les habilitats adquirides en la construcció de xarxes, programació de dispositius, i anàlisi de protocols seran de gran valor en el camp de les xarxes i les comunicacions.