UNIVERSITAT DE BARCELONA

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

Antoni Tuduri & Alejandro Guzman

2 desembre 2022

Índice

1. INTRODUCCIÓ

El model TCP/IP és una explicació de protocols de xarxa creat per Vinton Cerf i Robert E. Kahn, a la dècada de 1970. Va ser implantat a la xarxa ARPANET, la primera xarxa d'àrea àmplia (WAN), desenvolupada per encàrrec de DARPA , una agència (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.

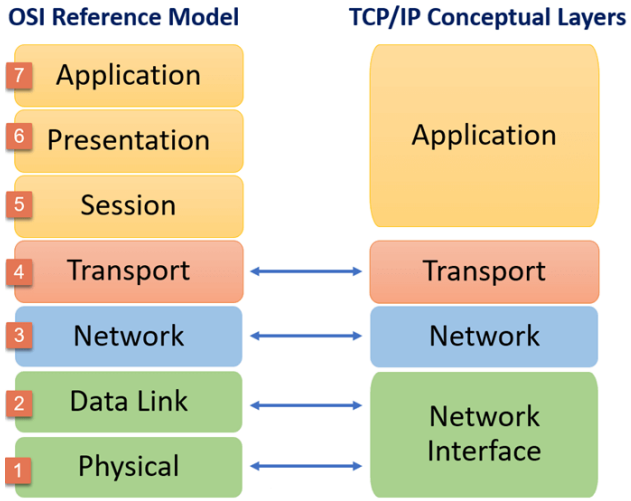

Figura 1.1: Relació model OSI i TCP/IP

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 vesant. 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 es 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 recol·lector 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 figura 2.1.

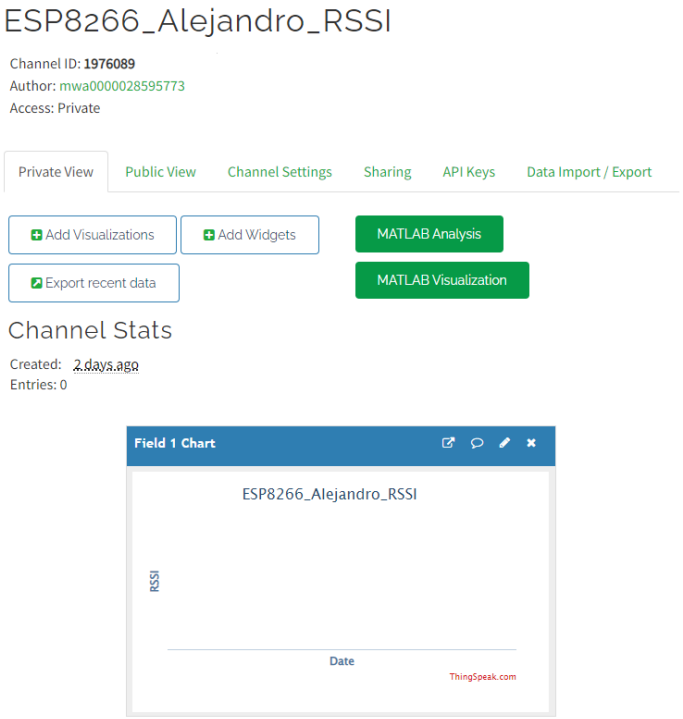

Figura 2.1: Canal de recollida de dades en Thingspeak

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 en la pràctica 2.

1.1. Firmware de la mota

Amb l'objectiu mencionat, dissenyarem un pretit programa en Arduino on implementarem un menú molt senzill amb el qual poder escanejar les xarxes WiFi disponibles, connectar-se a una i enviar a Thingspeak els valors RRSI obtingut de la WiFi conectada, amb possibilitat de realitzar

la desconnexió en finalitzar l'enviament de les dades.

Inicialment declararem una variable del tipus WifiClient el qual ens permet connectar-nos a thinkspeak 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 WiFi a la que estem connectats afegint-ho dins del paquet.

Per enviar el paquet de dades es necessari el write API Key per així poder escriure les dades al canal de Thinkspeak 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 figura 2.2.

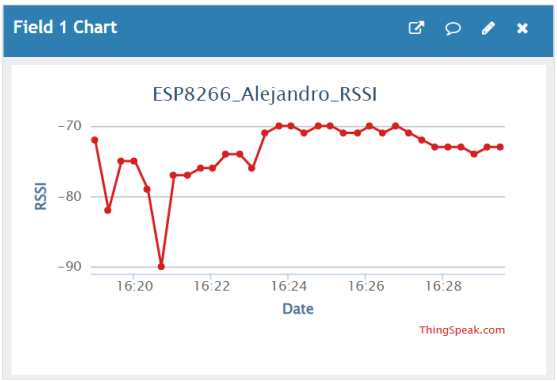

Figura 2.2: Canal de recollida de dades en Thingspeak actualitzat

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 WiFi amb la qual ens podem connectar i transmetre informació. Per crear un punt d'accés l'únic que haurem de fer es configurar la WiFi que ens aporta la llibreria com un Acces Point. Amb un altre mota podem veure que amb el codi de la practica anterior ens hem pogut connectar sense cap problema entre 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 missatges. Per tant necessitarem crear dos codis, un per a cada mota, primer configurem el servidor i posteriorment el client que es connectará a aquest servidor i enviarà dades.

Inicialment, en el nostre programa, cconfigurarem una placa en mode WIFI_STA(STATION) per a poder connectar-la a l'altre placa que haurà d'estar configurada en WIFI_AP(Soft ACCESS POINT) per a poder estalir la seva pròpia xarxa WiFi. D'aquesta manera estem inicialitzant una placa com a client i una altra com a servidor, respectivament.

```
#include <ESP8266WiFi.h>

char ssid[] = "MiFibra-F1C6";
char password[] = "vEtegk5w";

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 2.3: Configuració de la mota com a punt d'accés

3.1. Servidor

Primer configurem una de les dues motes com a servidor. Llavors utilitzarà el mode de WIFI_AP. La configuració de la placa és la que es mostra en la figura 2.4.

```
// 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 2.4: Configuració de la mota com a punt d'accés

A continuació inicialitzem la xarxa WiFi 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 l'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 finalitzar de construïr un chat 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 en la figura 2.5 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 2.5: 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 figura a 2.6 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 2.6: Mètode per enviar el missatge

4. Simulació amb Packet Tracer

L'última part de la pràctica té com a objectiu construïr 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.

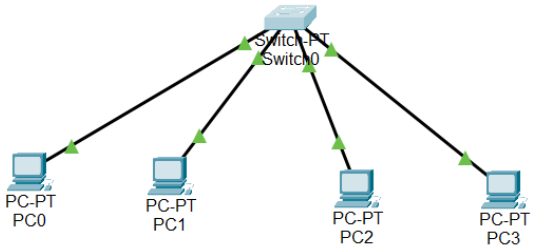

Figura 2.7: Esquema de les nostres connexions

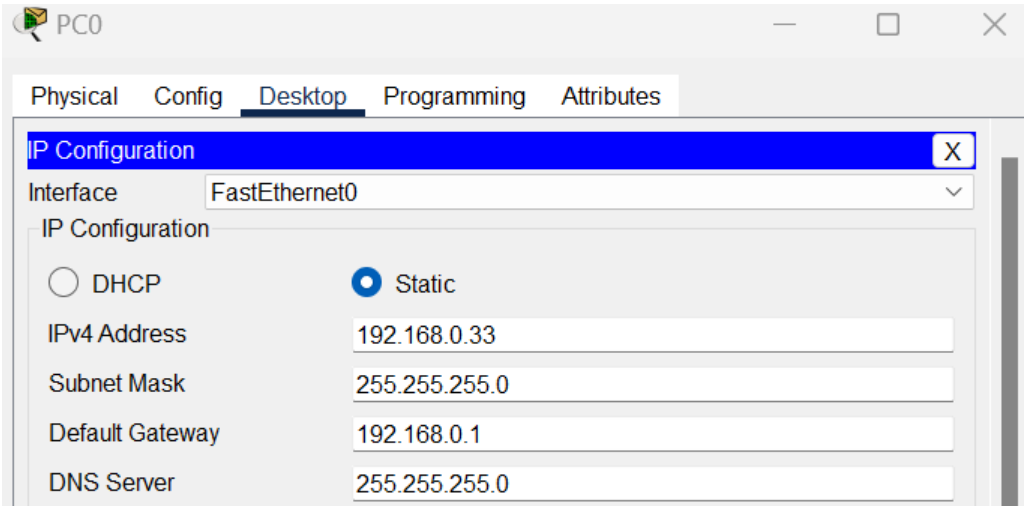

Figura 2.8: IP estàtica a cada un dels 4 PCs

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 2.9: PING al ordinador 1.

Com podem veure a la figura 2.10, que mostra la Event List, el que fa el PC des del qual fem ping és enviar un packet a tots els PC. El PC que rep el paquet en qüestió respon i aleshores el primer fa connexió amb el segon.

Event List

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 2.10: 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 el 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 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
```

Les dues imatges següents són les configuraciones del router 0. 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 rebi una sol·licitut d'un ordinador o servir, poder localitzar fàcilment la informació que volem en un altre router i establir una connexió de comunicació.

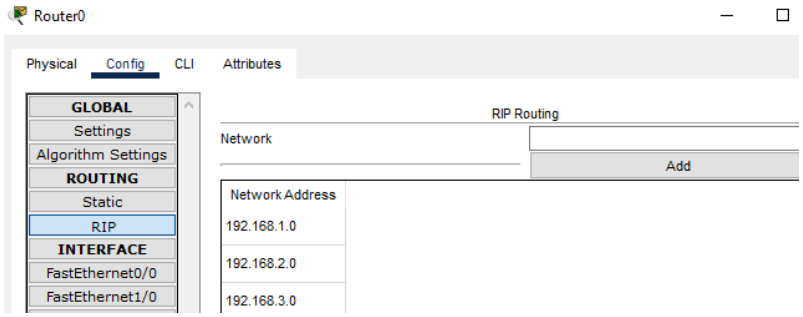

Figura 2.11: Router 0 RIP

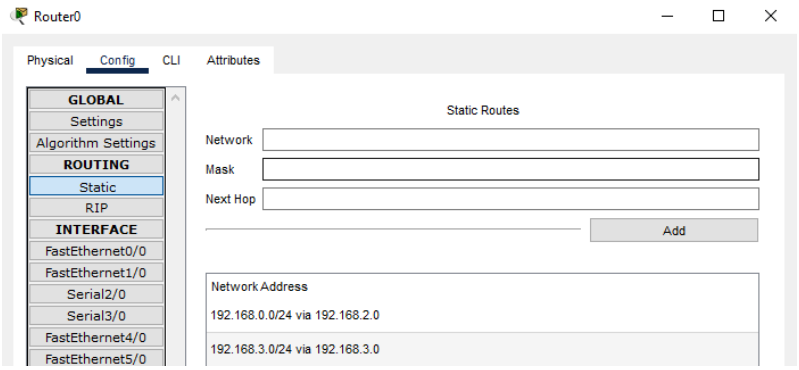

Figura 2.12: Router 0 static

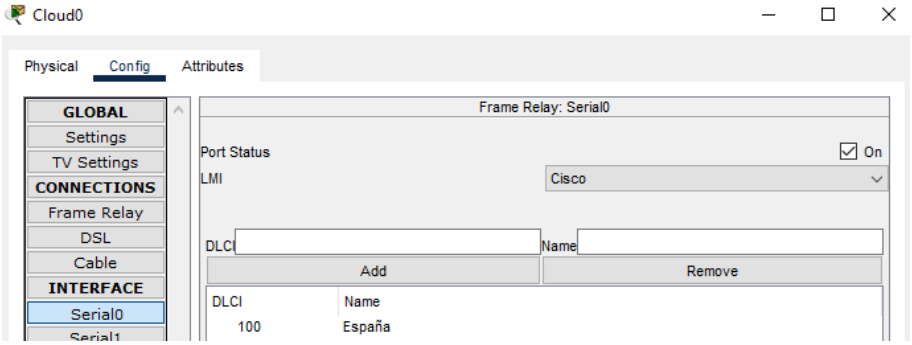

Figura 2.13: Serial 0

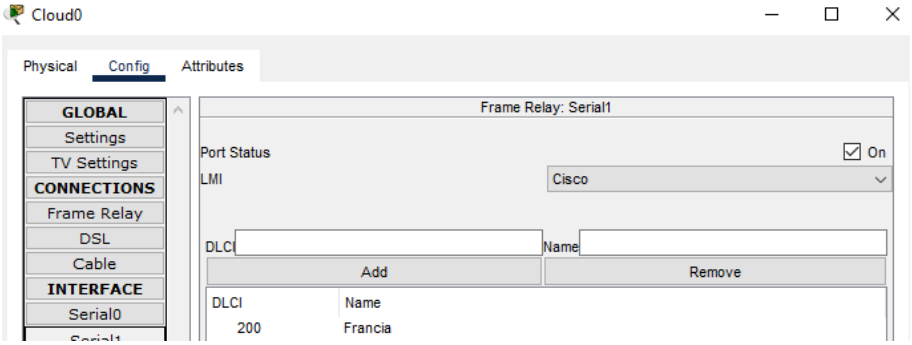

Figura 2.14: Serial 1

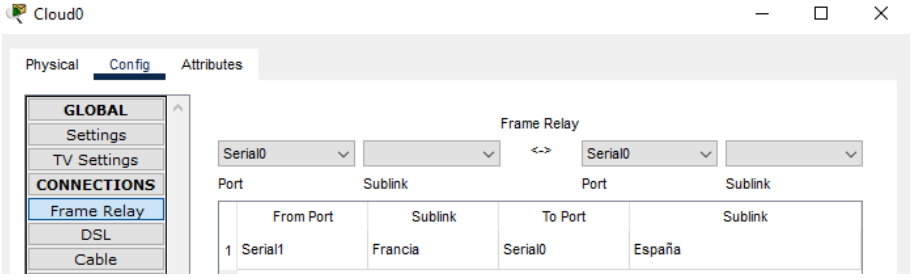

Figura 2.15: Frame Relay per connectar entre si

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 figura. A part podem veure que en el primer router també hi posem un sniffer, mitjançant el sniffer podem comprovar l'estat de la informació.

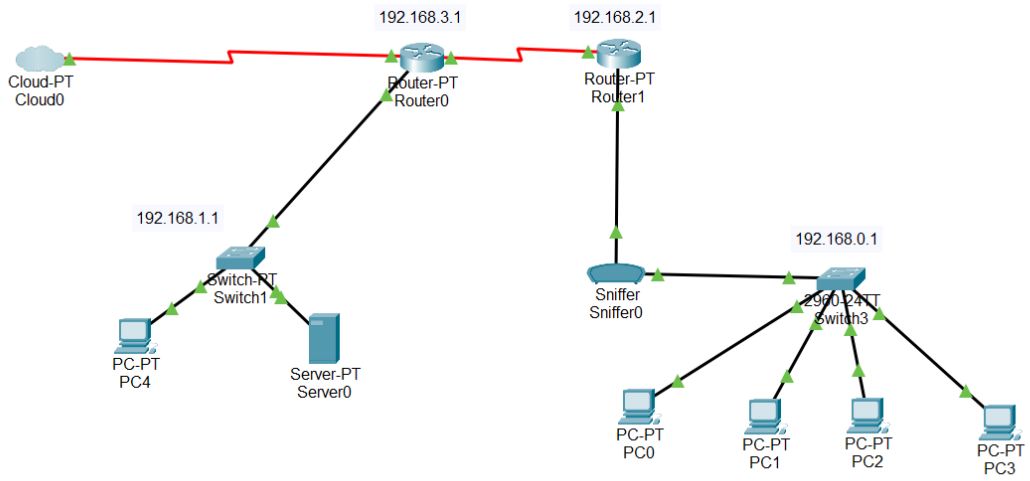

Figura 2.16: Topologia final de la WAN configurada

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

Figura 2.17: Server a PC

4.2. Sniffer i protocols de xarxa

Mitjançant 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 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.

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

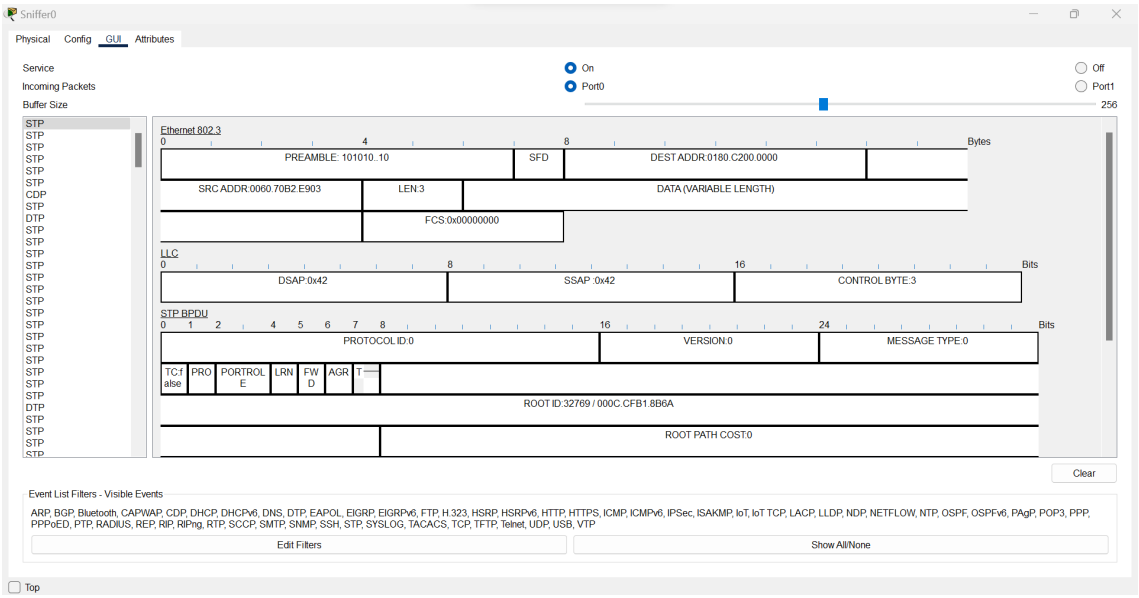

Figura 2.18: Sniffer i els seus protocols

3. CONCLUSIONS

Aquesta pràctica ens ha servit molt per a acabar d'aprofundir en els conceptes estudiats en la pràctica anterior, podent-los aplicar d'una forma més directa que com ho vam fer abans, a més a més ens sembla una manera interactiva i divertida de comprovar el funcionament de les xarxes.

A part hem combinat diversos programes per a aplicar els mateixos conceptes teòrics, lo qual ens ha ofert diversos punts de vista sobre els mateixos.