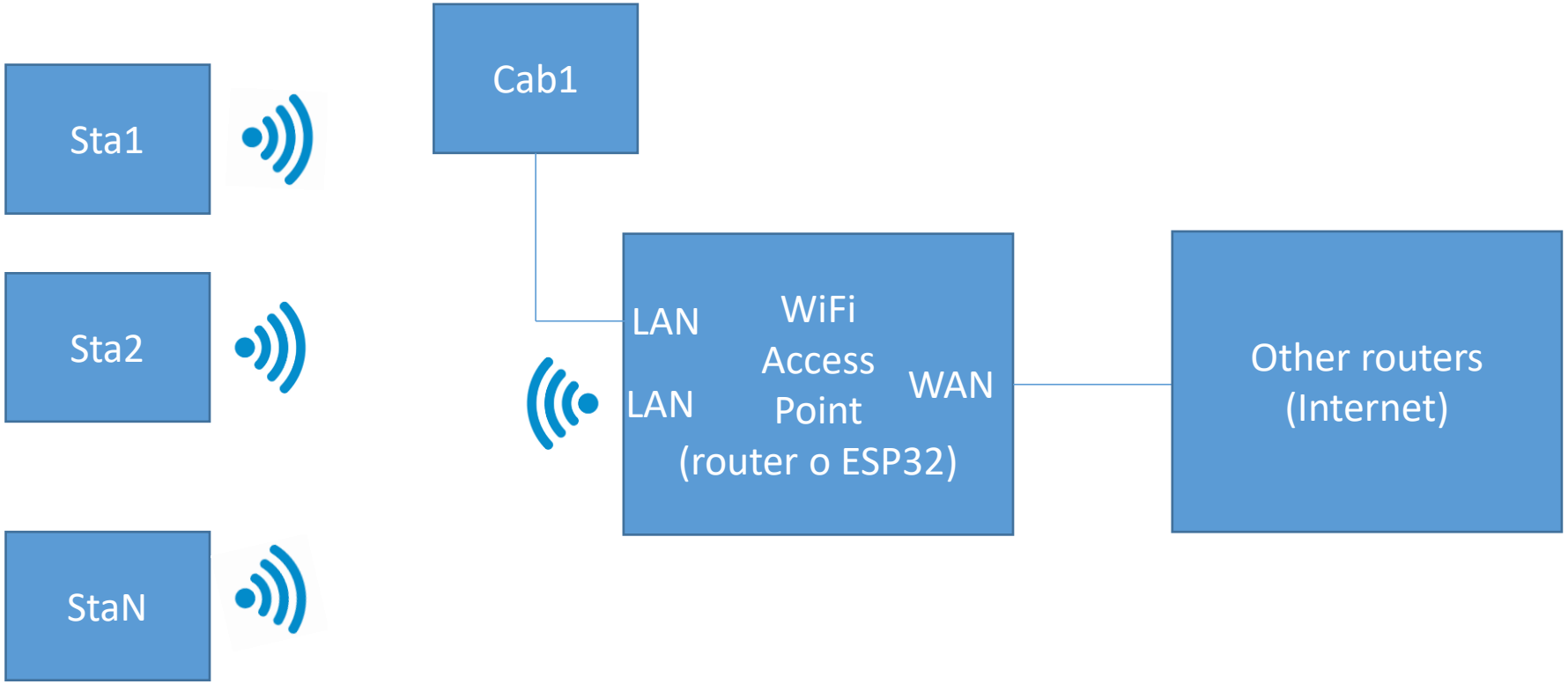
Comunicaciones TCP/IP en múltiples lenguajes

Ignacio Alvarez - Sept 2024



Access Point: - Establece nombre (SSID) y clave (passwd) a utilizar por las estaciones
- Determina rango direcciones IP en LAN
- Asigna dirección IP a estaciones si usan DHCP
- Espera solicitudes de conexión de estaciones

Stations: - Solicitan conexión a Access Point usando SSID y passwd
- Una vez aceptada, pueden usar las direcciones IP de la LAN para comunicación

Cabled: - Conexión directa, sin SSID ni passwd
- Pueden usar las direcciones IP de la LAN para comunicación

Configuración Router WiFi como punto de acceso:

- Conectar con cable
- Comprobar dirección IP asignada por el router con ipconfig (ej. **192.168.100**.242)
- Entrar en http://**192.168.100.**1
- Configurar según instrucciones del Router:
  - LAN: direcciones IP de la parte LAN (WiFi y cableadas), DHCP, permisos, etc.
  - WAN: acceso al exterior

ESP32/Arduino como punto de acceso (modo Soft AP):

- Define el SSID del AP y el password (ej MIM-GRUPO-X y grupo-x)
- Define los datos de la subred creada para los WiFi que se conecten (ej. dir ip 192.168.23.1)

ESP32/Arduino

```
#include <WiFi.h>
#include <WiFiClient.h>

void setup() {
  const char* ssid="MIM-GRUPO-X";
  const char* pwd="grupo-x";

  IPAddress local_ip(192,168,23,1);
  IPAddress gateway(192,168,23,1);
  IPAddress subnet(255,255,255,0);

  WiFi.softAP(ssid, pwd);         // Start Acces point mode
  WiFi.softAPConfig(local_ip, gateway, subnet);
  Serial.print("My IP: ");
  Serial.println(WiFi.softApIP());

  delay(100);
}
```

ESP32/Arduino como estación (STA):

- Debe conocer el nombre (SSID) y el password del router (ej MIM-GRUPO-X y grupo-x)
- Puede obtener datos de red (IP, etc.) de dos formas:
  - Dirección fija (debe conocer la subred del router, y usar una dirección en esa subred que el router no vaya a asignar por DHCP)
  - Dirección dinámica (asignada por el router, puede cambiar cada vez que nos conectemos a él)

ESP32/Arduino

```
#include <WiFi.h>
#include <WiFiClient.h>

void setup() {
  const char* ssid="MIM-GRUPO-X";
  const char* pwd="grupo-x";

  WiFi.mode(WIFI_STA);                  // Modo Station
  WiFi.begin(ssid, pwd);
  while (WiFi.status() != WL_CONNECTED)
    {    delay(1000);
         Serial.print(".");
    }
  Serial.println("Conectado");
  Serial.print("My IP: ");
  Serial.println(WiFi.localIP());
}
```

```
// Añadir si se quiere una IP fija (necesaria para servidor)
IPAddress local_ip(192,168,23,44); // Comprobar ajustes en router
WiFi.config(local_ip);
```

Servidor:

- Debe tener dirección IP fija y conocida, ejemplo 192.168.1.44
- Crear objeto y esperar conexión (ejemplo en puerto 55355)

Para comunicaciones IP dentro del mismo equipo: dirección 127.0.0.1 = localhost

Matlab	Python	ESP32/Arduino

`>> server=tcpserver(55355);`
`>> counter=0; while server.Connected==0`
`  fprintf('.'); pause(1);`
`  counter=counter+1;`
`  if mod(counter,40)==0`
`  fprintf('\n');`
`  end`
`end; fprintf('Conectado\n');`
`>> ...enviar o recibir ...`
	`>>> import socket`
	`>>> server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)`
	`>>> server.bind(('0.0.0.0', 55355))`
	`>>> server.listen(1)`
	`>>> conn=server.accept()`
	`>>> print('Conectado')`
	`>>> ...enviar o recibir ...`
	
	Ojo doble paréntesis necesario (es una tupla)
		`...`
		
		`WiFiServer server(55355);`
		`WiFiClient conn;`
		
		`void setup() {`
		`  ... Conexión de red (AP o STA) ...`
		`  server.begin();`
		`}`
		
		`void loop() {`
		`  if (! conn) {`
		`  conn=server.available();`
		`  if (conn)`
		`  Serial.println("Cliente conectado\n");`
		`  }`
		`  if (conn)`
		`  {`
		`  ...enviar o recibir ...`
		`  }`
		`}`

Qt-SDK
Descargar http://isa.uniovi.es/~ialvarez/2023-2024/Mecatronica/C3-ISC/Descargas/Comu_Sockets_con_Doc_doxygen.rar?version=0001 Ayuda en : Doc/html/index.html

C estándar
Ejemplo: http://isa.uniovi.es/~ialvarez/Descargas/pagina_de_prueba.htm Ayuda en : http://isa.uniovi.es/~ialvarez/Curso/descargas/ProgramarWinsockQt.pdf

Cliente:

- Debe conocer la IP y puerto del servidor
- Crear objeto y conectar con el servidor anterior

Ojo doble paréntesis necesario (es una tupla)

Matlab	Python	ESP32/Arduino

`>> cli=tcpclient('192.168.1.44',55355);`
`>> ...enviar o recibir ...`
	`>>> import socket`
	`>>> cli = socket.socket(socket.AF_INET, socket.SOCK_STREAM)`
	`>>> cli.connect(('192.168.1.44', 55355))`
	`>>> ...enviar o recibir ...`
		`WiFiClient cli;`
		
		`void setup() {`
		`  ... Conexión de red (AP o STA) ...`
		
		`}`
		
		`void loop() {`
		`  if (! cli.connected( )) {`
		`  cli.connect("192.168.1.44",55355);`
		`  if (cli.connected( ))`
		`  {`
		`  ...enviar o recibir ...`
		`  }`
		`}`

Qt-SDK
Descargar http://isa.uniovi.es/~ialvarez/2023-2024/Mecatronica/C3-ISC/Descargas/Comu_Sockets_con_Doc_doxygen.rar?version=0001 Ayuda en : Doc/html/index.html

C estándar
Ejemplo: http://isa.uniovi.es/~ialvarez/Descargas/pagina_de_prueba.htm Ayuda en : http://isa.uniovi.es/~ialvarez/Curso/descargas/ProgramarWinsockQt.pdf

Enviar texto (lo puede hacer tanto el cliente como el servidor, una vez conectados)

Matlab (xxx = server o cli)	Python (xxx = conn o cli)	ESP32/Arduino (xxx = conn o cli) Sustituir en *...enviar y recibir ...*
`>> text=sprintf('Hello\n');` `>> xxx.write(text);`	`>>> text='Hello\n'` `>>> xxx.send(str.encode(text))`	`String texto="Hello\n";` `xxx.write(texto);`

Recibir texto en forma bloqueante (lo puede hacer tanto el cliente como el servidor, una vez conectados)

Matlab (xxx = server o cli)	Python (xxx = conn o cli)	ESP32/Arduino (xxx = conn o cli) Sustituir en *...enviar y recibir ...*

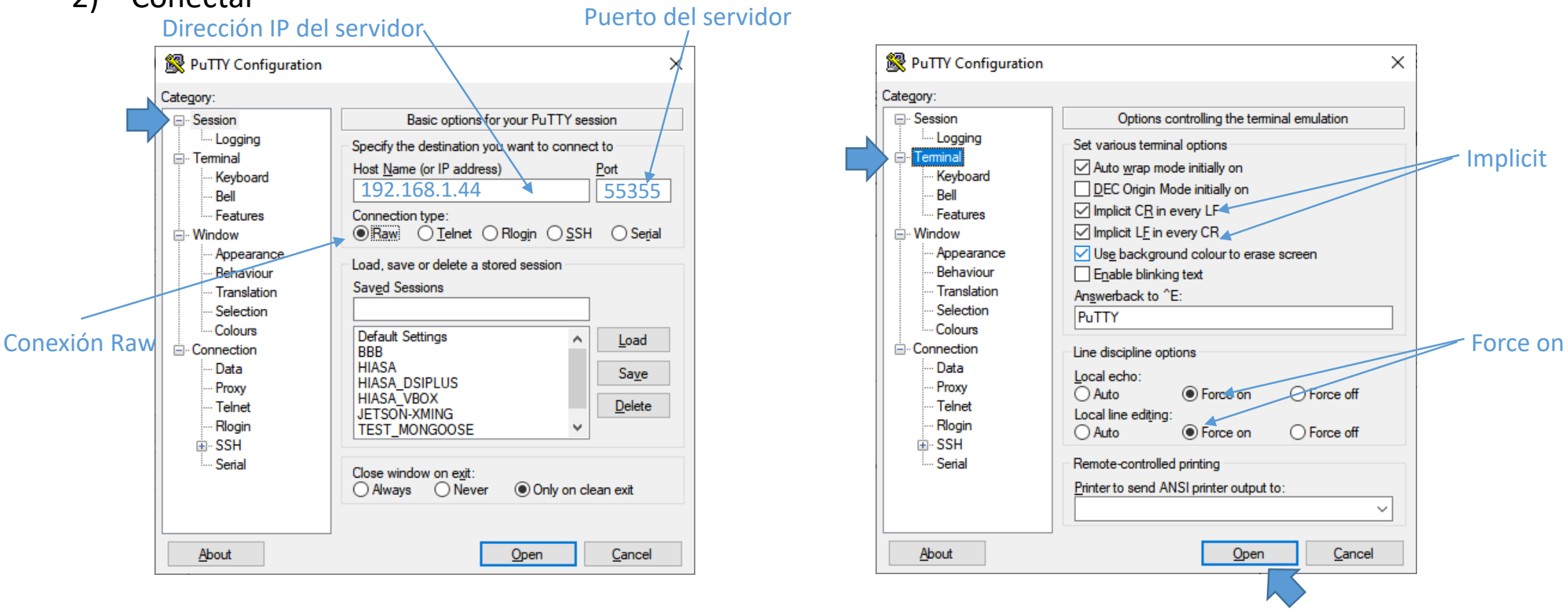
`>> while (xxx.NumBytesAvailable==0)`
`end`
`>> recv=char(xxx.read());`
	`>>> xxx.setblocking(True)`
	`>>> tam_max=1024`
	`>>> bytes=xxx.recv(tam_max)`
	`>>> recv=bytes.decode()`
		`if (xxx.available() ) {`
		`String recv=conn.readStringUntil('\n');`
		`Serial.print("RECV: ");`
		`Serial.println(recv);`
		`}`

Qt-SDK
Descargar http://isa.uniovi.es/~ialvarez/2023-2024/Mecatronica/C3-ISC/Descargas/Comu_Sockets_con_Doc_doxygen.rar?version=0001 Ayuda en : Doc/html/index.html

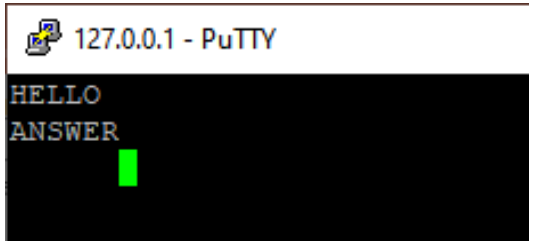
C estándar
Ejemplo: http://isa.uniovi.es/~ialvarez/Descargas/pagina_de_prueba.htm Ayuda en : http://isa.uniovi.es/~ialvarez/Curso/descargas/ProgramarWinsockQt.pdf

Cliente para pruebas con putty

1) Instalar putty desde https://www.putty.org/ (Windows) o con apt (Ubuntu)
2) Conectar



3) Escribir texto + INTRO para enviar, la recepción es automática



Servidor para múltiples clientes con ESP32, estilo Arduino:

- Mantiene una lista de clientes
- Cada uno debe identificarse con su nombre, tras recibir el mensaje: "<<<NAME>>>"
  (se envía automáticamente tras la conexión, el cliente debe responder con "<<<NAME=NombreQueIdentificaAlCliente>>>\n")
- A partir del nombre, el servidor puede decidir qué enviar/recibir de cada cliente

ESP32 (archivo .ino)

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <vector>

#include "MyServer.h"

void setup() {
  // ... Conectar WiFi SoftAP o STA ...
  _server.begin();
}

void loop() {
   CheckNewConn();
   int j=CheckReceive();
   if (j>=0) {
      printf("Mensaje recibido de %d (%s) = %s\n",
             j,_conn[j].name.c_str(),_conn[j].recv.c_str());
      SendTo(j,"Hello\n");
   }

   // Ejemplo: envía a todos un contador cada 1 segundo
   static uint32_t lastTimeSent=0,counter=0;
   uint32_t t=millis();
   if (t-lastTimeSent> 1000) {
      counter++;
      lastTimeSent=t;
      char msg[32];
      sprintf(msg,"Counter=%ld\n",counter);
      SendToAllStartingWith("",msg);
   }
}
```

MyServer.h

```
#include <Arduino.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <vector>

extern WiFiServer _server;  // servidor (nº de puerto definido en MyServer.cpp)
struct MyClient {   // estructura para cada cliente conectado
  WiFiClient cli;      // cliente conectado
  String name;         // nombre identificativo ("UNKNOWN" si no ha recibido)
  String recv;         // Ultimo mensaje recibido del cliente
} ;
extern std::vector<struct MyClient> _conn; // lista de clientes activos

// Comprueba si algún cliente nuevo se ha intentado conectar. En caso afirmativo,
// envía el mensaje "<<<NAME>>>\n" para que se identifique, y devuelve true
bool CheckNewConn();

// Comprueba si algún cliente ha enviado un mensaje. En caso afirmativo, devuelve el
// índice del cliente en la tabla _conn, y se puede obtener el contenido en el campo
// recv de dicha tabla. Si el mensaje es de identificación, asocia el nombre al
// cliente. Si no hay mensaje nuevo, o el mensaje era de identificación, devuelve -1.
int CheckReceive();

// Envía mensaje a un cliente a partir de su índice en la tabla (ojo, el índice puede
// cambiar al eliminarse elementos de la tabla, usar solamente tras CheckReceive)
int SendTo(int j,const char* txt);

// Envía mensaje a un cliente a partir de su nombre identificativo
int SendTo(const char* name,const char* txt);

// Envía mensaje a todos los clientes cuyo nombre identificativo empiece por name
// (si name es "", envía a todos)
int SendToAllStartingWith(const char* name,const char* txt);
```

Servidor para múltiples clientes con ESP32, estilo Arduino:

- Mantiene una lista de clientes
- Cada uno debe identificarse con su nombre, tras recibir el mensaje: "<<<NAME>>>"
  (se envía automáticamente tras la conexión, el cliente debe responder con "<<<NAME=NombreQueIdentificaAlCliente>>>\n")
- A partir del nombre, el servidor puede decidir qué enviar/recibir de cada cliente

MyServer.cpp

```
#include "MyServer.h"

WiFiServer _server(55355); // ¡¡¡ CAMBIAR POR NUM DE PUERTO DESEADO !!!
std::vector<struct MyClient> _conn;

bool CheckNewConn() {
   WiFiClient new_conn=_server.available();
   if (new_conn) {  // nuevo cliente ha solicitado conexión
      struct MyClient remote;
      remote.cli=new_conn;
      remote.name="UNKNOWN";
      int j=_conn.size();
      _conn.push_back(remote); // añade a lista
      IPAddress remoteIp=remote.cli.remoteIP();
      printf("Cliente %d conectado. Remote IP: %d,%d.%d.%d , port:%d\n",
             j, remoteIp[0],remoteIp[1],remoteIp[2],remoteIp[3],
             remote.cli.remotePort());

      // Solicita identificación al cliente
      char ans[]="<<<NAME>>>\n";
      printf("Enviando: %s",ans);
      _conn[j].cli.write(ans);
      return true;
   }
   return false;
}
```

MyServer.cpp (continúa)

```
int CheckReceive() {
    for (int j=0;j<_conn.size();j++) { // para todos los clientes
      if (_conn[j].cli.available() ) { // se ha recibido algo del cliente
        _conn[j].recv=_conn[j].cli.readStringUntil('\n'); // actualiza recv
        printf("Recibido cliente %d: %s\n",j,_conn[j].recv.c_str());
        char ans[64];
        if (_conn[j].recv.startsWith("<<<NAME=")) { // es identificación
            int end=_conn[j].recv.indexOf(">>>");
           _conn[j].name=_conn[j].recv.substring(8,end); // actualiza name
           _conn[j].recv="";
        }
        else if (_conn[j].name=="UNKNOWN") { // aún no identificado
           strcpy(ans,"<<<NAME>>>\n");       // repite solicitud
           printf("Enviando: %s",ans);
           _conn[j].cli.write(ans);
        }
        else {
           return j;  // Devuelve el nº del cliente
        }
      }
      else if (! _conn[j].cli.connected() ) { // Cliente desconectado
         printf("Cliente %d (%s) desconectado\n",j,_conn[j].name.c_str() );
         _conn.erase(_conn.begin()+j); // Elimina de la lista
      }
   }
   return -1;
}
```

Servidor para múltiples clientes con ESP32, estilo Arduino:

- Mantiene una lista de clientes
- Cada uno debe identificarse con su nombre, tras recibir el mensaje: "<<<NAME>>>"
  (se envía automáticamente tras la conexión, el cliente debe responder con "<<<NAME=NombreQueIdentificaAlCliente>>>\n")
- A partir del nombre, el servidor puede decidir qué enviar/recibir de cada cliente

MyServer.cpp (continúa)

```
int SendTo(int j,const char* txt) {
   return _conn[j].cli.write(txt);
}

int SendTo(const char* name,const char* txt) {
   for (int j=0;j<_conn.size();j++) {
      if (_conn[j].name==name) // El nombre concide
         return SendTo(j,txt);
   }
   return -1;
}


int SendToAllStartingWith(const char* name,const char* txt) {
   int count=0;
   for (int j=0;j<_conn.size();j++) {
      if (_conn[j].name.startsWith(name)) {// El nombre concide
         SendTo(j,txt);
         count++;
      }
   }
   return count;
}
```