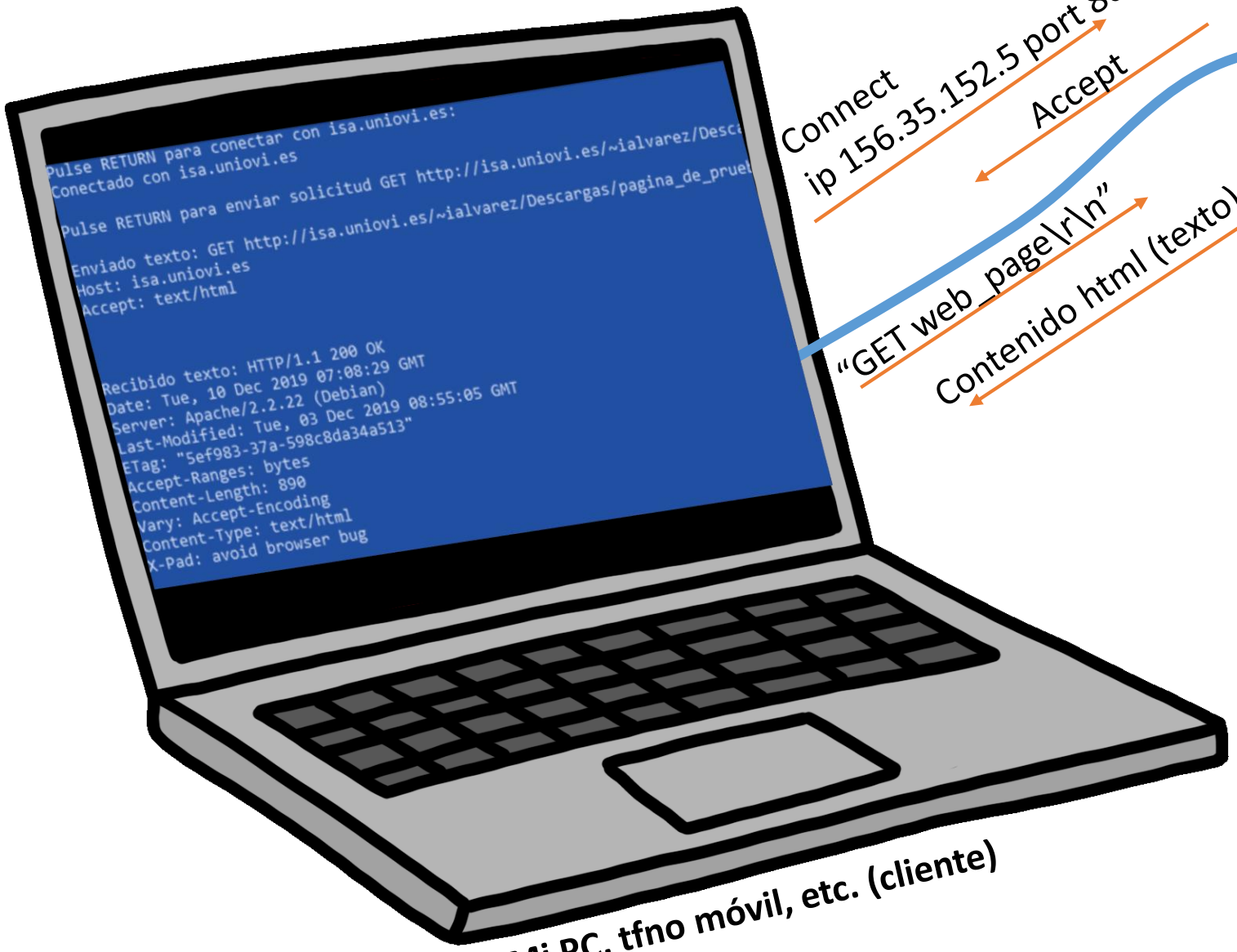
EJEMPLO COMUNICACIÓN TCP ENTRE PC y servidor páginas web



Connect
ip 156.35.152.5 port 80

Accept

"GET web_page\r\n"

Contenido html (texto)

Router

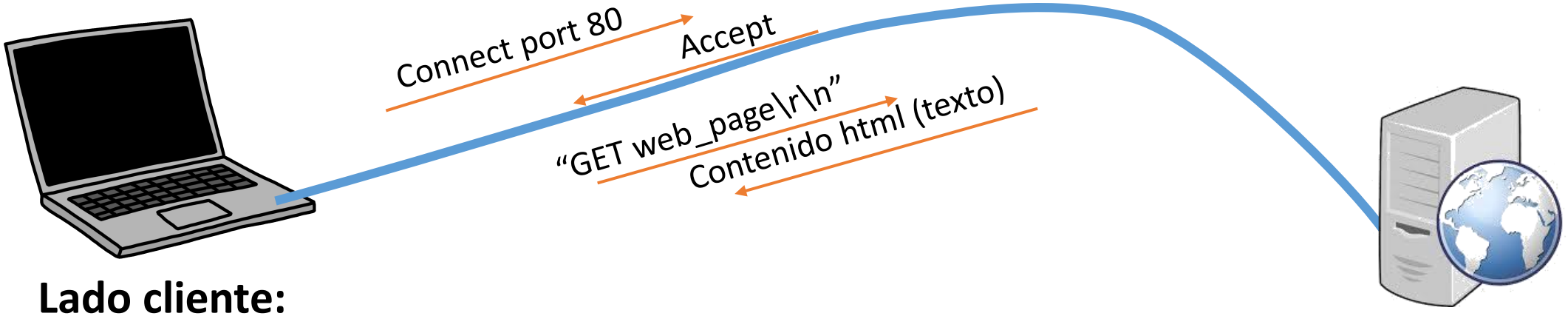
Router

Router

Mi PC, tfno móvil, etc. (cliente)

isa.uniovi.es (servidor)
ip = 156.35.152.5
port = 80 (http)

EJEMPLO COMUNICACIÓN TCP ENTRE PC y servidor páginas web



Lado cliente:

- Inicializa socket TCP (tipo stream) → id. ***sock***
- Enlaza (bind) ***sock*** con dirección IP y puerto local (cualesquiera)
- Intenta conexión TCP de ***sock*** con el remoto que responde en la dirección IP y puerto deseado del servidor
- Si se ha establecido conexión, sigue protocolo http :
  - Escribe mensaje "GET …" en dispositivo ***sock***
  - Lee respuesta del servidor recibida en ***sock***
  - Decodifica respuesta:
    - Determina que el contenido es text/html
    - Decodifica el texto html: si detecta un hyperlink en la respuesta, propone seguirlo
    - Si el usuario decide seguir el link, hace lo mismo para la nueva página
- Cierra ***sock***

Lado Servidor:

- Inicializa socket TCP (tipo stream) → id. ***server***
- Enlaza (bind) ***server*** con dirección IP 156.35.152.5 y puerto reservado para http (80)
- Espera conexiones de forma genérica con listen()
- Espera conexión específica con accept(). Cuando se produce:
  - Se ha creado un nuevo socket ***connected*** para esta conexión (***server*** queda liberado para un nuevo accept(), permitiendo conexiones simultáneas)
  - Espera y lee mensaje recibido en ***connected*** según protocolo http
  - Decodifica, lee archivos involucrados y genera respuesta según protocolo http
  - Envía respuesta escribiendo en ***connected***
  - Cierra ***connected*** (si el cliente no ha solicitado mantenerlo)

PROGRAMA PRINCIPAL PC (Windows) 1 de 2

```
// EJEMPLO DE APLICACION PARA OBTENER TEXTO DE UN SERVIDOR WEB (http)
// 1) Conecta con el servidor http (usar isa.uniovi.es), puerto 80
// 2) Prepara solicitud para la pagina:
//       http://isa.uniovi.es/~ialvarez/Descargas/pagina_de_prueba.htm
// 3) Envia la solicitud y recibe el texto html
// 4) Dhtml y busca enlaces a otras paginas
// 5) Si hay algun enlace, permite rehacer una solicitud para la pagina
//    indicada en el mismo y volver al punto 3

#include <stdio.h>
#include <stdlib.h>
#include <winsock2.h>

void EndProgram(const char* msg,int err);

int main()
{
    WSADATA wsaData;
    WORD wVersionRequerida = MAKEWORD (2, 2);
    SOCKET sock;
    struct sockaddr_in local,remoto;
    char txt_snd[1024],txt_rcv[65536];
    char txt_page[1024],txt_host[1024];
    int n;

    // Necesario en Windows para inicializar sockets para la aplicación
    WSAStartup (wVersionRequerida, &wsaData);

    // Obtener identificador para un socket TCP
    sock=socket(AF_INET,SOCK_STREAM,0);
    if (sock==INVALID_SOCKET)
    {
        EndProgram("No se puede crear socket",-1);
    }

    // Preparar estructura con datos de IP/puerto local y realizar bind()
    local.sin_family=AF_INET;
    local.sin_addr.S_un.S_addr=0;   // 0 = cualquier dirección IP disponible
    local.sin_port=htons(0);        // 0 = cualquier puerto disponible
    memset(local.sin_zero,0,8*sizeof(char));
    if (bind(sock,(struct sockaddr*) &local,sizeof(struct sockaddr_in))!=0)
    {
        printf("No se puede asignar dirección local. Programa terminado\n");
        return -2;
    }

    printf("Pulse RETURN para conectar con isa.uniovi.es: ");
    getchar();

    // Preparar estructura con datos de IP/puerto remoto y realizar connect()
    remoto.sin_family=AF_INET;
    remoto.sin_addr.S_un.S_addr=inet_addr("156.35.152.5");
    remoto.sin_port=htons(80);
    memset(remoto.sin_zero,0,8*sizeof(char));

    if (connect(sock,(struct sockaddr*) &remoto,sizeof(struct sockaddr_in))!=0)
    {
        EndProgram("No se puede conectar a servidor remoto",-2);
    }
    printf("Conectado con isa.uniovi.es\n\n");

    // Preparar campos iniciales para GET: página a visitar y servidor
    strcpy(txt_page,"http://isa.uniovi.es/~ialvarez/Descargas/pagina_de_prueba.htm");
    strcpy(txt_host,"isa.uniovi.es");

    char op;        // Opciones de finalizacion: 'S', 'N' ó 'F'

    for (op='S';op!='F';)  // Bucle principal, mientras no se seleccione 'F'
    {
        printf("Pulse RETURN para enviar solicitud GET %s :",txt_page);
        getchar();

        // Preparar texto de solicitud de página web
        strcpy(txt_snd,"GET ");
        strcat(txt_snd,txt_page);
        strcat(txt_snd, " HTTP/2.0\r\n");
        strcat(txt_snd,"Host: ");
        strcat(txt_snd,txt_host);
        strcat(txt_snd,"\r\nAccept: text/html\r\n\r\n");

        // Enviar texto al remoto
        n=send(sock,txt_snd,strlen(txt_snd),0);
        if (n!=strlen(txt_snd))
        {
            EndProgram("Error en envio",-3);
        }

        printf("Enviado texto: %s\n\n",txt_snd);
```

PROGRAMA PRINCIPAL PC (Windows) 2 de 2

```
// Esperar respuesta y guardar en txt_rcv
  for (n=0;n==0;)
      n=recv(sock,txt_rcv,65536,0); // (*) Ver mejor solución al final
  if (n<0) {
      EndProgram("Error en recepcion",-4);
  }
  txt_rcv[n]=0;   // Asegura que la cadena recibida termina en nulo
  printf("Recibido texto: %s\n\n",txt_rcv);
```

```
  // Obtener campos relevantes de respuesta
  int answerIsOk=0;
  const char *ptType=GetContentsAfter(txt_rcv,"Content-Type:");
  if (ptType!=NULL) {
     if (strncmp(ptType,"text/html",strlen("text/html"))==0)
        answerIsOk=1;
  }
  if (! answerIsOk) {
      EndProgram("Falta Content-Type en respuesta",-5);
  }

  int len=-1;
  const char* ptLen=GetContentsAfter(txt_rcv,"Content-Length:");
  if (ptLen!=NULL)
      len=atoi(ptLen);
  if (len<0) {
      EndProgram("Falta Content-Length en respuesta");
  }
  char *ptStart,*ptEnd;   // ptStart=lugar donde comenzar a buscar,
                          // ptEnd=fin de busqueda
  ptStart=strstr(txt_rcv,"\r\n\r\n");
  if (ptStart!=NULL) {
      ptStart=strstr(ptStart+4,"<!DOCTYPE html");
      if (ptStart==NULL) {
          EndProgram("Pagina no contiene documento html",-6);
      }
      ptStart=strchr(ptStart,'>')+1;
  }
```

```
     // Bucle para buscar enlaces html (cadenas en el texto recibido con el
        // formato <a href="direccion_del_enlace">Texto</a>)

        for (;ptStart!=NULL;) // Bucle busca enlaces
        {
            ptStart=strstr(ptStart,"<a href=");
            if (ptStart!=NULL)  // Se ha encontrado un enlace
            {
                ptStart+=strlen("<a href=")+1;
                ptStart++;      // Apunta al siguiente caracter tras las
                                // comillas de comienzo de dirección de enlace
                ptEnd=strchr(ptStart,'>');
                ptEnd--;        // Apunta a las comillas de final de dirección
                                // de enlace
                n=ptEnd-ptStart;  // Num de caracteres entre las comillas
                strncpy(txt_page,ptStart,n);   // Modifica la pagina a visitar
                txt_page[n]=0;  // Asegura terminación en carácter nulo

                printf("Referencia a: %s\n",txt_page);
                printf("Ir a pagina? <S/N/F> (Si/No/Fin): ");

                for (op='X';op!='S' && op!='N' && op!='F';)
                {
                    op=toupper(getchar());  // op = siempre mayuscula
                }
                getchar();  // Elimina \n del buffer de teclado
                if (op=='S' || op=='F')
                    break;
                if (op=='N')
                    ptStart=ptEnd+1;
            }   // Fin if encontrado enlace
        } // Fin Bucle busca enlaces
    } // Fin de Bucle principal, mientras no se seleccione 'F'

     // Fin del bucle principal: cerrar socket y aplicación

    closesocket(sock);
    printf("Fin del programa. Pulse INTRO para terminar ");
    getchar();

    return 0;
}
```

PROGRAMA PRINCIPAL PC (Windows) (fn aux)

```
// (*)
// Solución para la recepción en que los datos del servidor pueden llegar troceados en varios paquetes
// Se esperan datos del servidor durante un tiempo máximo, y se van añadiendo al destino si llegan troceados
// Invocar con n=WaitAndReceive(sock,txt_rcv,65536); en lugar de n=recv(…);

int WaitAndReceive(SOCKET sock,char* dest,int tam_dest) {
    ULONG NonBlock = 1;

    if (ioctlsocket(sock, FIONBIO, &NonBlock) == SOCKET_ERROR)  // Socket en modo no-bloqueante
    {
        printf("ioctlsocket() failed with error %d\n", WSAGetLastError());
        fflush(stdout);
        return -1;
    }

    int n_recv,total_len_recv;
    for (total_len_recv=0;total_len_recv<tam_dest;)  // Seguir recibiendo paquetes mientras no se produzca timeout
    {
        FD_SET rd;
        struct timeval tout={ 0,500000 };     // Time-out: 0 s + 500000 us=500 ms
        FD_ZERO(&rd);
        FD_SET(sock,&rd);
        if (select(1,&rd,NULL,NULL,&tout)>0)    // Comprueba si en sock se ha recibido algo; termina cuando se recibe o vence el timeout
        {
            n_recv=recv(sock,dest+total_len_recv,tam_dest-total_len_recv,0);  // Como máximo caben tam_dest-len_recv
            if (n_recv>0)
                total_len_recv+=n_recv;    // Acumulamos
            else
                break;
        }
        else
            break;
    }

    return total_len_recv;
}
```