Redes de Computadoras

Solución – 26 de julio de 2018

(ref: solredes20180726.odt)

Instrucciones

- **Indique su nombre completo y número de cédula en cada hoja.**
- **Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado y utilice una caligrafía claramente legible.**
- **Comience cada pregunta teórica y cada ejercicio en una hoja nueva.**
- **Sólo se responderán dudas de letra. No se responderán dudas de ningún tipo los últimos 30 minutos del examen.**
- **El examen es individual y sin material. Apague su teléfono celular mientras esté en el salón del examen.**
- **Es obligatorio responder correctamente al menos 15 puntos en las preguntas teóricas y 20 de los problemas prácticos. Los puntos ganados en el curso se suman a los puntos de teórico.**
- **El puntaje mínimo de aprobación es de 60 puntos.**
- **Para todos los ejercicios, si es necesario, puede suponer que dispone de los tipos de datos básicos (p.ej. lista, cola, archivo, string, etc.) y sus funciones asociadas (ej: tail(lista), crear(archivo), concatenar(string, string).**
- **Justifique todas sus respuestas.**
- **Duración: 3 horas. Culminadas las 3 horas el alumno no podrá modificar las hojas a entregar de ninguna forma.**

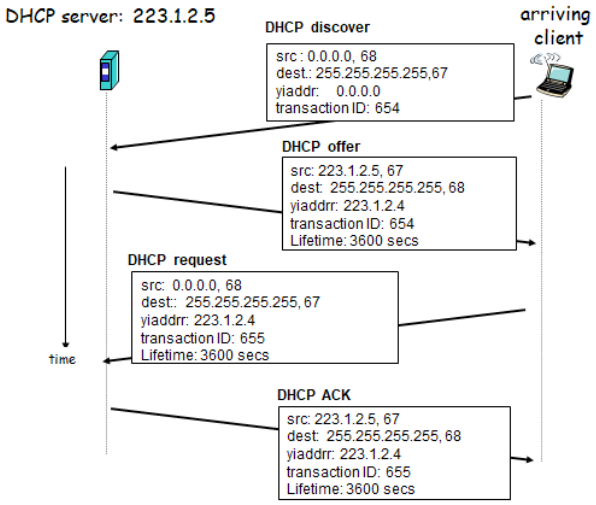
Preguntas Teóricas

Pregunta 1 (4 puntos)

a) Resuma la utilidad del protocolo DHCP.
b) Describa los dos mensajes utilizados para la obtención de los parámetros de configuración de la red del cliente.

Solución Pregunta 1

a) El protocolo DHCP, Dinamic Host Configuration Protocol, es una de las formas disponibles para que un nodo conectado a una red IP obtenga una dirección desde un servidor DHCP, el que gestiona un pool de direcciones IP a tales efectos. Las dos opciones para que a un nodo se le asigne una dirección IP son: que se configure estáticamente ("hardcoded") en la propia configuración de red del nodo, o que la obtenga al unirse a la red, por ejemplo, mediante DHCP. Con este protocolo logramos mayor flexibilidad en el uso del rango de direcciones IP disponibles para numerar los nodos de una red, pudiendo reutilizarlas, especialmente en redes donde los nodos no requieren estar siempre conectados y con la misma dirección IP, como sería el caso de los servidores. Adicionalmente, DHCP puede entregar más información, como por ejemplo, nombre y dirección IP de un servidor DNS.



b) Figura adjunta. De los 4 mensajes del handshake, DHCP discover, DHCP offer, DHCP request y DHCP ack, se deben describir los últimos dos: Request y Reply (ACK).

Pregunta 2 (6 puntos)

Discuta la utilidad del atributo AS-PATH de BGP, y comente si es una buena métrica para elegir un NEXT-HOP para el tráfico hacia un determinado AS.

Solución Pregunta 2

El atributo AS-PATH contiene la lista de ASs que ha atravesado la publicación de un prefijo. Cuando un prefijo es pasado a un AS, éste agrega su ASNumber al atributo. Este atributo tiene dos usos fundamentales: 1) evitado de loops en las publicaciones (si un AS se encuentra a sí mismo en el AS-PATH); 2) si es tenido en cuenta en el proceso de selección de ruta y cuando se lo considere (existen otros atributos), se seleccionará aquella que tenga el AS-PATH más corto.
La utilización del AS-PATH como métrica de selección del NEXT-HOP puede no ser la mas adecuada si se busca un camino de costo mínimo. Esto se debe a que no se conoce la cantidad de saltos (hops) por los que se atraviesa dentro de cada AS.

Pregunta 3 (10 puntos)

a) Explique en qué consiste la característica de *store and forward* (almacenamiento y reenvío) de los *switches* (conmutadores) de capa de enlace.
b) ¿Qué ventajas tiene para la capa de red que la capa de enlace utilice este mecanismo, en comparación a tener una red puramente conectada con hubs?

Solución Pregunta 3

a) Consiste en que las comunicaciones de datos se hacen en unidades de transferencia (frames en el caso de capa de enlace), las cuales deben ser recibidas en su totalidad (store) antes de poder ser reenviadas (forward) o consumidas. Este mecanismo es bien diferente al utilizado por los hubs, los que reenvían inmediatamente por todos los puertos cada bit (en lugar de frames) que reciben.

b) Dado que cada frame contiene no solo los datos, sino también información de control; luego de obtenido el frame completo se podrían implementar controles y procesos locales que permiten servicios de chequeos de correctitud, entregas confiables, etc.
El reenvío selectivo y autoaprendizaje también son ventajas que derivan directamente del procesamiento luego de recibido el frame completo y que son una de las mejoras notorias en el rendimiento entre switches y hubs.

Pregunta 4 (8 puntos)

a) Suponga que debe configurar una red en la que solo existe un único camino ruteado posible entre cualquier par de routers. Defina si utilizaría un protocolo de ruteo dinámico o estático, argumentado en favor de su decisión.
b) Explique las diferencias en términos de robustez y convergencia para los algoritmos vector distancia y estado del enlace.
c) ¿Qué tipo de algoritmo utilizaría en una red donde hay cambios frecuentes en los costos de los enlaces?

Solución Pregunta 4

a) En una red con forma de árbol utilizaría ruteo estático, ya que la ausencia de caminos alternativos hace que no tenga sentido mantener un proceso (que además puede fallar) que resuelve una situación que no se dará nunca (seleccionar otro camino ante la caída de un enlace).

b) Los algoritmos de estado de enlace (LS) comunican costos de enlaces, y cada nodo computa los costos mínimos independientemente de los demás, mientras que en vector de distancias (DV) se comunican (y propagan) los costos de los caminos al resto de los nodos, iterando sobre estos cambios. Por lo tanto un fallo en DV puede potencialmente afectar a toda la red, mientras que en LS el efecto está acotado.
Además, en LS como cada nodo posee una visión de toda la topología de la red, puede rápidamente calcular un nuevo camino ante el cambio de costo de algún enlace.

c) En este caso utilizaría un algoritmo de estado enlace. El dato de que muchos enlaces pueden cambiar sus costos sería muy malo para un protocolo de vector de distancias, ya que "las malas noticias viajan lento" y eso llevaría a períodos de convergencia largos con frecuencia. Sin embargo, se debe tener en cuenta que, en estado de enlace, si la red es muy grande, el costo de mantener la info de la red para los dispositivos sería importante.

Pregunta 5 (12 puntos)

a) ¿Por qué es necesario utilizar números de puerto en las comunicaciones de capa de transporte, si a nivel de capa de red y utilizando solo direcciones de red es posible comunicar un host de origen con uno de destino? ¿cuál es el nombre que se le da a esta comunicación entre estas dos entidades en TCP?
b) Tomando como ejemplo el protocolo TCP, ¿por qué es necesario utilizar timers? Mencione dos ejemplos en los que se evidencia su necesidad.
c) Explique conceptualmente las diferencias entre control de flujo y control de congestión; ¿qué eventos son los que generan transiciones/acciones en la FSM de control de congestión de TCP?

Solución Pregunta 5

a) Porque la necesidad real es comunicar dos procesos lógicos y no solo dos dispositivos, que además pueden existir más de uno en el mismo dispositivo por lo cual es necesario multiplexar/demultiplexar con mayor granularidad que solo la dirección de red. A la comunicación entre dos procesos lógicos en TCP se le llama conexión.

b) Existen situaciones particulares en las cuales el emisor y el receptor esperan el uno por el otro sin tomar acción alguna, haciendo necesario destrabar esa situación con otro mecanismo. Supongamos que un emisor no tiene nada más para transmitir, y el receptor recibió los segmentos pero se perdieron las confirmaciones, en este caso el emisor reenviará su ventana luego de vencido el timeout.

c) El control de flujo tiene sentido en una conexión entre un origen y un destino, mientras que el control de congestión intenta responder al estado general de la red entre ese origen y destino, adaptándose a ese estado ajeno a los extremos de esa conexión. Los eventos interesantes para el control de congestión son la llegada de un ACK, la llegada de un ACK triplicado y el vencimiento del timer.

Problema 1 (30 puntos)

Una empresa contrata una conexión a Internet a un proveedor que instala un router con una configuración que deja la red pública 190.1.1.0/29 accesible en la interfaz de red local de dicho router con dirección IP 190.1.1.1. Dicho router es administrado por el proveedor de Internet (es decir, su configuración NO se puede cambiar), y permite acceso completo desde/ hacia Internet a la red pública. La empresa tiene dos servidores DNS (maestro y esclavo), aloja su página web y además pretende ofrecer un servicio de streaming sobre UDP, balanceando carga entre dos o más servidores físicos. La empresa está organizada en las secciones Comercial, Administración y Gerencia, que tienen 25, 31 y 6 usuarios respectivamente, pero es posible que crezcan un 60% c/u. Todos los usuarios deben tener acceso a Internet y a los servidores públicos de la empresa, pero no deben poder acceder al resto de las secciones.

Se pide:

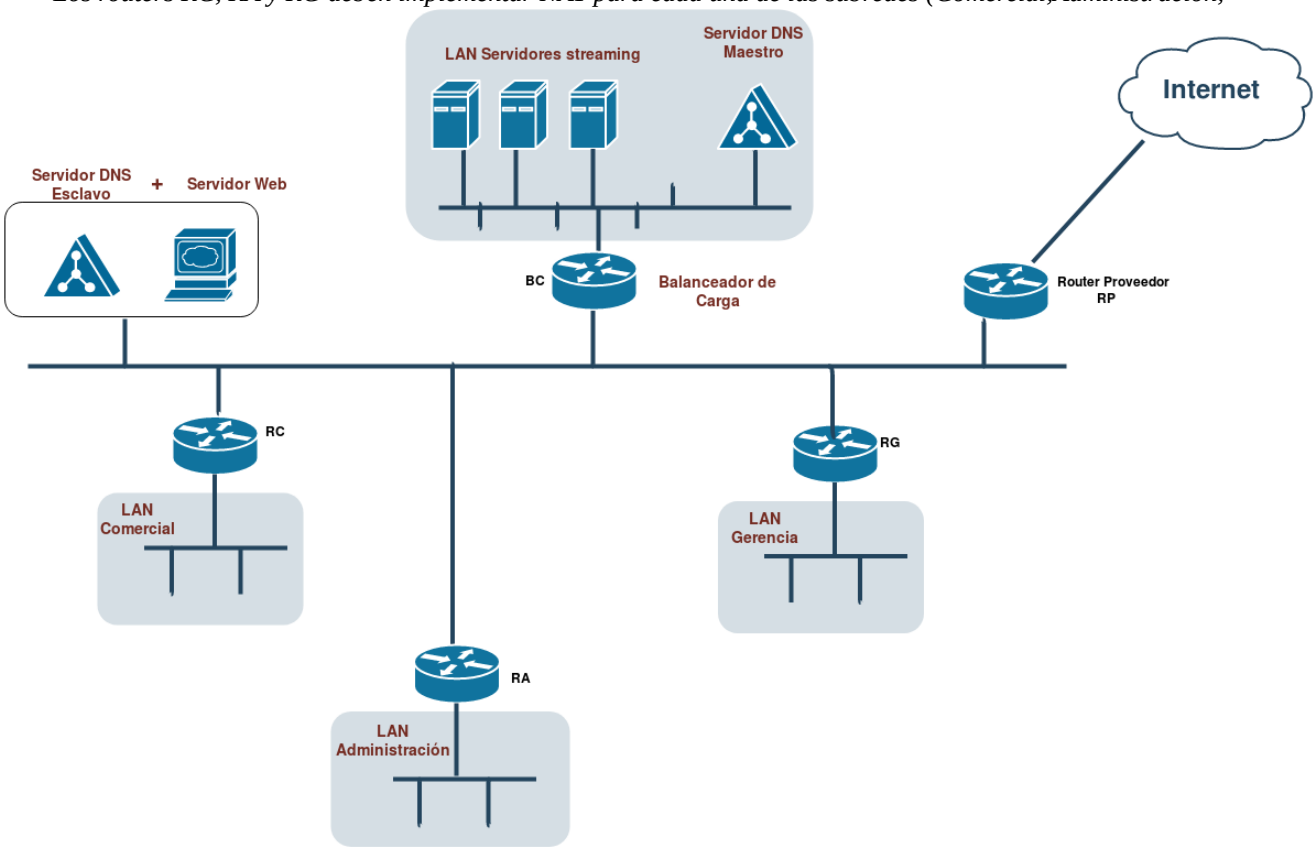
a) Proponga una arquitectura de red para cumplir con los requerimientos. Especifique los dispositivos a agregar y las funcionalidades que deben tener. Presente la solución propuesta en un diagrama.

b) Asigne direcciones IP para todos los sistemas de la empresa, utilizando direcciones IP privadas en caso de ser necesario. La asignación debe ajustarse estrictamente a las necesidades especificadas.

c) Configure las tablas de forwarding de todos los dispositivos para cumplir con los requerimientos. Utilice el formato `Prefijo de Destino | Interfaz de salida | Next-Hop.`

Solucion Problema 1
a)
Se presenta la propuesta de solución de la figura:

- *Los routers RC, RA y RG deben implementar NAT para cada una de las subredes (Comercial,Administración, Gerencia); las tablas de forwarding deben configurarse adecuadamente para cumplir con los requerimientos (ver parte c).*



- *El balanceador de carga para el servicio de streaming BC debe tener algún algoritmo de balanceo (por ejemplo algo simple como round-robin), y debe mantener las sesiones entre los usuarios y el servidor elegido de forma transparente al usuario. A su vez debe distinguir solicitudes al servicio DNS Maestro.*
- *DNS Esclavo y el servidor Web de la empresa comparten equipo.*

b)

LAN pública 190.1.1.0/29
RP: 190.1.1.1
Servidor DNS Esclavo y Web: 190.1.1.2
BC: 190.1.1.3 (Es compartida con el servicio DNS Maestro)
RC: 190.1.1.4
RA: 190.1.1.5
RG: 190.1.1.6

*LAN Comercial: 25*1.6 → 40 hosts.*
Se necesitan además: 1 dir IP para el RC, dirección de red y broadcast, 43 direcciones IP en total → prefijo /26.

*LAN Administración: 31*1.6 → 50 hosts.*
Se necesitan además: 1 dir IP para el RA, dirección de red y broadcast, 53 direcciones IP en total → prefijo /26.

*LAN Gerencia: 6*1.6 -> 10 hosts*
Se necesitan además: 1 dir IP para el RG, dirección de red y broadcast, 13 direcciones IP en total → prefijo /28.

Dado que las tres subredes son estancas (es decir, solo se comunican con "el exterior" mediante NAT pero no entre si), se puede usar el mismo prefijo de red, por ejemplo se puede asignar:

LAN Comercial: 192.168.1.0/26
LAN Administración: 192.168.1.0/26
LAN Gerencia: 192.168.1.0/28

Para evitar confusiones y permitir futuros cambios de configuración se pueden asignar prefijos diferentes, por ejemplo:

LAN Comercial: 192.168.1.0/26
LAN Administración: 192.168.2.0/26
LAN Gerencia: 192.168.3.0/28

Para la LAN de Servidores de streaming y Servidor DNS Maestro se puede utilizar cualquier prefijo porque no se imponen restricciones, por ejemplo 10.0.0.0/8.

En todos los casos se asigna la dirección de la LAN de direcciones privadas .1 a la interfaz de los routers y el balanceador.

c)

DNS Y WEB:

Prefijo de Destino	*Interfaz de salida*	*Next-Hop*
190.1.1.0/29	*eth0 (pública)*	*Directamente conectado*
0.0.0.0/0	*eth0 (pública)*	*190.1.1.1*

BC:

Prefijo de Destino	*Interfaz de salida*	*Next-Hop*
190.1.1.0/29	*eth0 (pública)*	*Directamente conectado*
0.0.0.0/0	*eth0 (pública)*	*190.1.1.1*
10.0.0.0/8	*eth1 (privada)*	*Directamente conectado*

Routers de subredes Comercial, Administración y Gerencia: solo deben tener configuradas las redes directamente conectadas y la ruta por defecto. NO se deben configurar rutas a las otras subredes; por este motivo se utilizan tres routers y no uno solo, ya que en ese caso no se podría resolver mediante el enrutamiento la aislación entre subredes.

RC:

Prefijo de Destino	*Interfaz de salida*	*Next-Hop*
190.1.1.0/29	*eth0 (pública)*	*Directamente conectado*
0.0.0.0/0	*eth0 (pública)*	*190.1.1.1*
192.168.1.0/26	*eth1 (privada)*	*Directamente conectado*

RA:

Prefijo de Destino	*Interfaz de salida*	*Next-Hop*
190.1.1.0/29	*eth0 (pública)*	*Directamente conectado*
0.0.0.0/0	*eth0 (pública)*	*190.1.1.1*
192.168.2.0/26	*eth1 (privada)*	*Directamente conectado*

RG:

Prefijo de Destino	*Interfaz de salida*	*Next-Hop*
190.1.1.0/29	*eth0 (pública)*	*Directamente conectado*
0.0.0.0/0	*eth0 (pública)*	*190.1.1.1*
192.168.3.0/28	*eth1 (privada)*	*Directamente conectado*

Todos los sistemas de las subredes RC, RA, RG y los servidores streaming tendrán en su tabla de forwarding una entrada de la subred local directamente conectada, y una ruta por defecto cuyo next-hop es la dirección IP .1 de cada subred. Todos estos sistemas tienen una sola interfaz de red (por ejemplo eth0).

Problema 2 (30 puntos)

Se desea permitir la visualización de partidos de fútbol en vivo en la red local de una empresa, utilizando metodologías de *streaming* de video, buscando que no se sature el enlace a Internet. El ancho de banda requerido por un *streaming* para la visualización de un partido es de 4 Mbps, y la empresa cuenta con una conectividad a Internet de 60 Mbps.

Para la implementación se propone desplegar una aplicación en un equipo local (*local-stream-server*), el cual recibe el *streaming* del partido deseado y lo re-transmite a todos los equipos que lo soliciten en la red local, como muestra la figura. La entrega del *streaming* se realiza reenviando todos los datos recibidos, a todos los equipos que lo hayan solicitado previamente.

El protocolo utilizado por los clientes para obtener un *stream* consiste en solicitar el inicio del envío sobre una conexiónTCP de control (puerto 554). Una vez que se especifica el comando adecuado sobre esta conexión, el servidor copia el *stream* que recibe desde el servidor *fifa-stream-server.com* mediante UDP en el puerto **<RECIBE>** al cliente correspondiente en el puerto especificado **<CLIENTE>**. La sintaxis se presenta a continuación:

- Para solicitar un *stream*, el cliente envía el comando
  STREAM <CLIENTE>\n
  sobre la conexión TCP de control, donde **<CLIENTE>** es el puerto UDP donde recepcionará el mismo.
- Para finalizar el *stream*, el cliente envía el comando
  STOP\n
  sobre la conexión TCP de control.

Se pide:

a) Implemente en alto nivel el servicio propuesto, ejecutado en el servidor *local-stream-server.empresa.com*. Se evaluará el uso de sockets e implementación de la solución propuesta.
b) Implemente en alto nivel el programa **`void ver_stream()`** que solicita y reproduce el *stream*. Asuma que los datos recibidos se muestran utilizando la función **`void mostrar_video(char* stream)`** y que se cuenta con la función bloqueante **`void finalizar()`** que indica cuando el usuario desea terminar la reproducción.

Obs: el servidor de *fifa-stream-server.com* está siempre transmitiendo el *stream* hacia *local-stream-server.*

Solución Problema 2

a)
Se propone generar la aplicación que en un thread atiende el puerto de control, generando un thread para cada cliente. La recepción y envío del stream se realiza en el main de la aplicación. El envío se realiza desde mismo socket para todos los clientes, enviando el mismo datagrama para cada cliente, cambiando el destino del datagrama UDP.
Se propone una solución completa en lenguaje C. No se espera el nivel de detalle expresado en esta solución.
Para mayor claridad se evitan los controles de errores.

```
// Guarda una lista de estuctura que tiene un socket address adentro
void *lista_conectados(CustomAddr *);
struct CustomAddr {
  struct sockaddr_in *address;
};

const RECIBE_UDP = PORT; // puerto a donde envía fifa-stream-server el stream
const RECIBE_TCP = 554;  // puerto de control de la aplicación

// Funcion principal
int main(int argc , char *argv[]) {

  // creo thread que espera las conexiones de control al 554
  pthread_create(&proceso_control);

  // socket utilizado para recibir el STREAM desde Internet
  int fd_fifa_recive = socket (AF_INET, SOCK_DGRAM, 0);

  //Configuracion del socket
  //se asocia el socket al puerto RECIBE_UDP
  struct sockaddr_in fifa_recive;
  fifa_recive.sin_family = AF_INET;
  fifa_recive.sin_addr.s_addr = INADDR_ANY;
  fifa_recive.sin_port = htons(RECIBE_UDP);

  bind(fd_fifa_recive,(struct sockaddr *)&fifa_recive, sizeof(fifa_recive));

  // crea socket para generar la retransmisión del STREAM
  int fd_envia_udp = socket (AF_INET, SOCK_DGRAM,0);

  // Se asume que el trafico solo viene de fifa
  // Para cada datagrama recibido, lo envía a todos los clientes
  char buff[1024];
  while(recv(fd_fifa_receive, buff, 1024, 0)) {
     for(cliente in lista_conectados) {
       // Envio a todos los clientes,
       // utilizando el socket UPD fd_envia_udp con destino socket_udp
       sendto(fd_envia_udp, buff, 1024, 0, &cliente.address,
              sizeof(cliente.address));
     }
  }
  close(fd_fifa_receive);
}


void *proceso_control(){

  // crea socket para generar control de protocolo de STREAM
  // se genera para la IP pública y en puerto  RECIBE_TCP
  int fd_control = socket (AF_INET, SOCK_STREAM, 0);


  struct sockaddr_in server_addr;
```

```
  server_addr.sin_family = AF_INET;
  server_addr.sin_addr.s_addr = INADDR_ANY;
  server_addr.sin_port = htons(RECIBE_TCP);

  bind(fd_control, (struct sockaddr *)&server_addr, sizeof(server_addr));

  listen(fd_control);

  int client_socket;
  struct sockadd_in client_addr;
  c=sizeof(struct sockaddr_in);
  while(client_socket = accept(fd_control, &client_addr, &c)) {
     // se genera un thread para cada cliente activo
     pthread_create(&administracion_cliente, client_socket, client_addr);
     }
  }
  close(fd_control);
}


void *administracion_cliente(int client_socket, sockaddr_in client_addr) {
  char buff[2000];
  char *comandos;
  CustomAddr address;
  address.sin_addr.s_addr = client_addr.in_addr.s_addr;

  //asumo que todo lo necesario lo puedo leer con 2000 bytes
  while (recv(client_socket, buff, 2000, 0) > 0){
     comandos = buff.split(' '); // parseo comandos
     // Si recibo STREAM Nro_puerto
     if (comandos[0] == 'STREAM') {
       // En el segundo parametro obtenemos el puerto
       address.sin_port = comandos[1];
       lista_conectados.add(address);
     }
     // Si recibo STOP
     if (comandos[0] == 'STOP') {
       lista_conectados.remove(addres);
       close(client_socket);
     }
  }
}
```

b)

Se propone un proceso que genera un thread para la recepción y visualización del stream recibido, mientras que el control se realiza en el programa principal (main).

```
const IP_SERVER = IP; // Ip del local-stream-server
const PORT_SERVER = 554; // Puerto de control del local-stream-server
const LOCAL_PORT = 12345;

// Proceso que maneja la conexion con cada cliente
void ver_stream() {

  // Establezco socket para control TCP 554
  // se genera para la IP pública y en cualquier puerto
  int fd_control = socket (AF_INET, SOCK_STREAM, 0);

  int fd_local_stream = connect(fd_control, IP_SERVER, PORT_SERVER);

  // Establezco socket para recibir STREAM
  int fd_visor = socket(AF_INET, SOCK_DGRAM, 0);

  //Configuracion del socket
  //se asocia el socket al puerto donde se recibe el video
  struct sockaddr_in local_recive;
  local_recive.sin_family = AF_INET;
  local_recive.sin_addr.s_addr = INADDR_ANY;
  local_recive.sin_port = htons(LOCAL_PORT);

  bind(fd_visor,(struct sockaddr *)&local_recive, sizeof(local_recive));

  // Guardo una referencia al proceso de visualizacion para finalizarlo
  int hijo_fd;
  hijo_fd = pthread_create(&process_visualizacion, fd_visor);

  // Envio STREAM puerto \n
  char* msg = "STREAM " + LOCAL_PORT + "\n";
  int msg_size = strlen(msg);
  send(fd_local_stream, msg, msg_size, 0);

  finalizar(); //funcion bloqueante hasta finalizar la visualización

  msg = "STOP\n";
  msg_size = strlen(msg);
  write(fd_local_stream, msg, msg_size);

  exit(hijo_fd); // cierra process_visualizacion()
  close(fd_local_stream);
  close(fd_visor);
}



void *process_visualizacion(fd_visor) {
  char buff[1024];
  while(recv(fd_visor, buff, 1024, 0) > 0) {
     mostrar_video(buff);
  }
}
```