Objetivo de la práctica.

Construcción de un subnivel sobre el nivel 1, que permita una mayor independencia de las aplicaciones clientes con los procesos de generación de tramas a nivel MAC. Los servicios de este nuevo nivel, que denominaremos nivel a, se accederán a través de una interfaz de programación de aplicaciones (*Application Programming Interface*, API) de comunicaciones. Este subnivel accederá a los servicios de red utilizando las funciones de nivel 1proporcionadas por la librería PCAP.

En esta práctica independizaremos los servicios/comprobaciones de recepción de los de emisión, de forma que cualquier comprobación sobre una trama de recibida se hará sobre el buffer de recepción y nunca dentro del bucle del programa principal.

El entorno de trabajo

La práctica se desarrolla sobre un entorno de comunicaciones Ethernet. Para ello se hará uso de la librería PCAP que nos permitirá enviar y recibir paquetes a bajo nivel.

Introducción

Esta práctica tiene por objeto construir un nuevo nivel jerárquico en nuestra red que permita aislar el programa de aplicación de los detalles específicos de construcción delas tramas y de las comprobaciones de error realizadas a nivel de transmisión. Puesto que no implementará todas las funciones propias de un nivel de enlace (nivel 2), lo denominaremos nivel1a.

Los servicios que proporcionará este nivel son los siguientes:

- Envío y recepción de tramas Ethernet sin confirmación (modelo datagrama}.
- Detección de la dirección origen de la comunicación.
- Composición y desmembramiento de las tramas de comunicaciones a partir de variables de programa. El cliente del nivel a enviará y recibirá sólo el campo de datos de la trama Ethernet, y la dirección MAC de la estación que se encuentra al otro extremo de la comunicación, siendo este nivel el que se encargará de componer la trama completa a partir de los datos que almacena.
- Verificación de las longitudes máxima y mínima de transmisión y realización de las acciones necesarias para su cumplimiento.
- Detección de errores mediante la generación y verificación de CRC.

Práctica

Ejercicio 1 Obligatorio: (4 puntos) Realizar un módulo en C que implemente el conjunto de primitivas de servicio que constituirán el nivel. Este módulo, se compondrá de dos archivos:

- nivelEth.c, con la implementación.
- nivelEth.h, con las declaraciones de las funciones, para su inclusión por el resto de los programas que utilicen el módulo, y todas las constantes y definiciones de tipo que sean necesarias para su correcta utilización.

Este módulo se deberá incluir como componente del resto de los programas de esta práctica y las siguientes en el correspondiente Makefile que los genere.

Como material de la práctica se entregará dos programas de ejemplos sobre los que el alumno deberá completar la práctica.

Ejercicio 2 Obligatorio: (2 puntos) Realizar un sencillo sistema de coloquio multiusuario.

Cada mensaje de chat se transmitirá por la red mediante una trama de Nivel Ethernet, en la que se incluirán los siguientes datos:

- Un campo de tamaño fijo de 20 caracteres, que indicará el tiempo en ASCII en el que se generó el mensaje (“DD:MM:AAAA HH:MM:SS “).
- Un campo de tamaño fijo de 20 caracteres, que indicará el nombre del canal de coloquio para el que se transmite el mensaje. En caso de que el nombre del canal sea más corto de 20 caracteres, se rellenará con caracteres nulos (0x0}.
- Un campo de tamaño fijo de 20 caracteres, que indicará el nombre del usuario que envía el mensaje. En caso de que el nombre del canal sea más corto de 20 caracteres, se rellenará con caracteres nulos (0x0).
- Un entero corto (uint16_t ,2 bytes) que indicará la longitud del campo siguiente.
- Un campo de longitud variable, que contendrá el texto del mensaje enviado al canal de coloquio.

La implementación de este sistema se realizará mediante dos programas: Un programa receptor de mensajes, y un programa emisor.

El programa receptor recibirá a su entrada un parámetro, una palabra de un máximo de 20 caracteres que será el nombre del canal de conversación del que se desea recibir la comunicación. Tras su arranque, quedará a la espera de recibir tramas de Tipo 2 que contendrán mensajes del canal. Cada vez que se reciba una trama, la función de notificación analizará su contenido, y si es un mensaje para el canal de coloquio cuyo nombre se pasó como parámetro en el arranque, notificará al programa principal la recepción de la trama. El programa principal una vez recibida la notificación de recepción correcta de trama la presentará en pantalla:

- El nombre del usuario que transmitió el mensaje.
- La dirección MAC de la estación de la red desde la que lo transmitió.
- El texto del mensaje recibido.
- Tiempo desde que se inició el chat (para obtener el momento de inicialización del chat, se puede utilizar la función `gettimeofday`).
- El tiempo transcurrido desde que se envió el mensaje.

```
/*****************************************************************/

/* Ejemplo gettimeofday() •/

/*****************************************************************/

#include <sys/time.h>

#include <stdlib.h>

#include <stdio.h>
int main(void) {
      struct timeval currentTime;

      gettimeofday(&currentTime,NULL};

      printf("Tiempo actual Secs:%1d,
MicroSecs:%1d\n",currentTime.tv_sec,currentTime.tv_usec);
      return OK;
}
```

Los modos de notificación de la función de recepción de tramas pueden ser mediante el cambio de valor de una variable global, o utilizando semáforos para la sincronización de los procesos de recepción y emisión.

De este modo continuará operando indefinidamente hasta que el usuario lo finalice pulsando Ctrl-C.

El programa emisor recibirá a su entrada dos parámetros por la línea de comandos: Una palabra de un máximo de 20 caracteres, que será el nombre del canal de conversación del que se desea recibir la comunicación; y una palabra de un máximo de 20 caracteres, que será el nombre del usuario que envía los mensajes desde dicho programa emisor. Tras esto, pedirá al usuario que teclee un texto, y lo transmitirá en difusión como una trama de tipo 2. De este modo continuará operando indefinidamente hasta que el usuario lo finalice pulsando Ctrl-C.

Implementar los programas solicitados, y realizar pruebas de su funcionamiento correcto con y sin errores de comunicaciones.

Ejercicio 3 Obligatorio: (2 puntos) Modificar el programa de coloquio para que remita un número determinado de tramas las tramas con campos de redundancia específicos para calcular su eficiencia.

Se modificarán las fuentes para crear 3 parejas de ejecutables para creación y comprobación de distintos cálculos de redundancia según las funciones disponibles.

- Checksum
- CRC16 bits
- CRC32 bits

Cada pareja de ejecutables, emisor/receptor, enviarán un número de mensajes, tramas, a definir, estos mensajes seguirán el mismo formato del ejercicio anterior con un mensaje formado por una cadena de caracteres fija.
El programa receptor, recibirá estos mensajes evaluando el código de redundancia de cada uno de ellos contando aquellas tramas que se han recibido correctamente, se consideran tramas correctas aquellas cuyo cálculo del campo de redundancia confirma que no se ha modificado la trama.
Al finalizar el programa se deberá mostrar el ratio de tramas perdidas.

Para analizar el comportamiento de dichos métodos en entornos de perdida de paquetes similares utilizando la herramienta `netem` de Linux con el siguiente formato:

- Para generar un 5% de error en la salida de los paquetes
  - `sudo tc qdisc add dev eth0 root netem corrupt 5%`
- Cambiar la tasa de error a un 10%
  - `sudo tc qdisc change dev eth0 root netem corrupt 10%`
- Borrar la tasa de error definida en la interfaz
  - `sudo tc qdisc del  dev eth0 root netem corrupt 5%`

Para realizar este estudio se generará un entorno de pruebas común para todas las simulaciones generando un entorno de prueba con una tasa de fallo concreta.

Para cada modelo, se reenviaran un total de 200 mensajes, de longitud 1500bytes con valores aleatorios entre 0-255, para, en recepción, evaluar la tasa efectiva de recepción medida como número de tramas correctas/total de tramas.

Cuestiones a responder justificadamente.

Ejercicio 4 Obligatorio: (2 puntos)

- Para cada algoritmo CRC (Checksum, CRC16,CRC32) representar el porcentaje de tramas perdidas frente a la tasa de error para los siguientes porcentajes de error.{0,5,10,15,25,50,75,90}
- ¿Cuál es la diferencia principal encontrada entre los distintos CRCs implementados?

- Explica la función del campo tipo en la trama Ethernet y su utilidad para implementar un stack de protocolos.

Documentación a Entregar

Se entregará el programa encargado de señalizar la práctica. Memoria de la práctica. Esta memora ha de incluir las respuestas y gráficas de las cuestiones teóricas comentadas anteriormente.

Memoria, programa y listados se entregarán vía web según el procedimiento establecido en las prácticas

Anexo I

Especificaciones de la interfaz de nivelEth

Los servicios de este nuevo nivel se accederán mediante una serie de funciones primitivas del nivel1a, cuyas especificaciones son las siguientes:

- ```
  typedef int (*tpfNotificacionRecepcionEth) (const uint8_t *, int, const
  uint8_t *, uint16_t, const struct timeval *);
  ```

- ```
  int InicializarEth(uint16_t *Tipos, uint8_t nType,
  tpfNotificacionRecepcionEth funcion, int timeout);
  ```

- ```
  int FinalizarEth(void);
  ```
  ```

- `int EnviarDatagramaEth(const uint8_t *direccion_destino, const uint8_t *mensaje, uint16_t tamano,uint16_t tipo);`

Además existirá, como en el nivel1 Ethernet, una función de notificación definida por el usuario, que será llamada cuando se reciba un datagrama válido.

La estructura timeval representa la marca temporal de la recepción de una trama. Esta marca estará en tiempo Unix, es decir en segundos desde el 1de enero de 1970 a las 00:00:00 UTC (era Unix). La estructura tiene dos campos:

- long int tv_sec, que representa los segundos (enteros) desde la era Unix.
- long int tv_usec, que representa en microsegundos el resto del tiempo transcurrido desde la era Unix.

```
int lnicializarEth(uint8_t Tipol , uint8_t Tipo2,
tpfNotificacionDesdeNivella Funcion,int Timeout);
```

ACCIÓN: Iniciación del nivelEth para su empleo por la aplicación.

- Inicia el nivel inferior (PCAP) de la interfaz (nivel 1) para permitir la recepción de tramas del tipo 1 (Ethertype 4444), del tipo 2 (Ethertype BBBB), o de ambos.
- Determina la dirección de la estación en la que estamos trabajando. Esta dirección se almacenará, y se empleará para la posterior construcción de las tramas. Esta dirección se obtendrá mediante la llamada a la función getMACAddr.
- Almacena el puntero a la función de notificación desde nivel a {que se recibe como parámetro) en una variable global, para su uso posterior en la función de notificación desde el nivel ,en dicha función de notificación se recibirán cada una de las tramas.
- Activa una variable global que indique que el nivel a ha sido inicializado correctamente.

PARÁMETROS:

int *Tipos: Array con los valores type definidos para la recepción de trama(0x4444, oxBBBB) o las deshabilitan (valor=NULL)

tpfNotificacionDesdeEth Funcion: Puntero a la función de notificación desde el nivelEth que se definirá en el nivel de aplicación.

int Timeout: Representa el periodo de tiempo, en milisegundos utilizado como timeout de lectura de tramas. Se pasará directamente al inicial el nivel 1 (PCAP).

VALOR DE RETORNO:

En el caso de que la operación se realice sin fallos, deberá poner la variable global Nivel1aIniciado (interna del nivel 1a) igual a 1 y devolver OK (0). En caso de fallo deberá devolver ERROR (-1).

La función de notificación desde el nivel Ethernet:

Puesto que esta función debe iniciar el nivelEthernet, la función de notificación desde el nivel 1 deberá pertenecer a este nuevo nivel 1a, es decir, deberá incluirse su código en este módulo. En esta función es, por tanto, la función donde se realizarán todas las comprobaciones necesarias para decidir si la trama es admitida o no, en función del protocolo que se definirá posteriormente.

Para implementar esta función se va a añadir a la trama de datos que se maneja en nuestro programa una sencilla Secuencia de Verificación de Trama (SVT). Esta secuencia consistirá en un código de redundancia cíclica a definir en cada modulo (Checksum, CRC16, CRC 64). Para ello se ha incluido como anexo una función genérica para el cálculo de CRCs. El proceso que se debe seguir es básicamente similar tanto al enviar como al recibir tramas. Por un lado, en transmisión se calcula el CRC de todo el contenido de la trama y se añade al final de la misma. Por otro lado, en recepción se debe calcular el CRC de toda la trama (incluyendo el CRC generado en transmisión). Una útil propiedad de los códigos de redundancia cíclica es que si la trama se ha transmitido correctamente obtendremos siempre un mismo valor, independientemente de los contenidos del mensaje. A este valor se le llama residuo, y para nuestro código es 0.

Deberá modificarse para que, después de comprobar que el CRC recibido es correcto, llame a la función de notificación desde el nivel 1a, siendo en esta última donde se decidirá finalmente si la trama debe ser aceptada o rechazada (según el código de retorno que proporcione). No debe olvidarse incluir la marca temporal de trama (struct timeval *) en la llamada a la función de notificación desde el nivel 1a. Esta llamada se realizará siempre a través del puntero a ella recibido como parámetro en la función IniciarNivel1a, y que fue almacenado en una variable global al realizarse la iniciación del nivelEth.

En caso de que el CRC recibido no sea correcto, la función de notificación desde el nivel 1 siempre descartará la trama, sin llamar a la función de notificación desde el nivel1a.

int FinalizarEth(void)

ACCIÓN: Esta función sólo llamará a la función de finalización de nivel 1 (PCAP) si NivelEthIniciado es igual a 1. Si esta última función devuelve 0, pondrá NiveEthIniciado igual a 0 y devolverá OK. Si hay error deberá escribir un mensaje "Error en nivel a" y salir con -1.

PARÁMETROS: No tiene.

VALOR DE RETORNO:

- OK (O): Operación realizada correctamente.
- ERROR (-1): La operación falló.

int EnviarDatagramaEth(const uint8_t *Dirección_Destino, int Tamano, const uint8_t

***Mensaje, int Tipo)**

ACCIÓN: Esta función envía un datagrama a otra estación. En el caso de que Nivellalniciado esté a 1, deberá formar la trama completa y llamar a EnviarTramaNivel1.

PARÁMETROS:

- const uint8_t *Dirección_Destino: Dirección de la estación destino.
- int Tamano: Longitud en bytes que se desea transmitir, es decir, la longitud de datos almacenada en el puntero Mensaje.
- const uint8_t *Mensaje: Puntero a la zona de memoria donde se encuentran los datos que se desean transmitir.
- int Tipo: Indica el tipo de trama que se envíar:
    - (Ethertype 0x8686)
    - (Ethertype 0x40AA)

VALOR DE RETORNO:

- ETH_OK (O): Operación realizada correctamente.
- ETH_ERROR (-1): La operación falló.

Como extensión a este nivel, igual que en el caso del nivel 1,el programa de aplicación deberá proporcionar una función auxiliar, que, por tanto, deberá estar programada dentro de alguno de los módulos que compongan el programa de aplicación, y no en el módulo que contiene la interfaz de nivel a. El nivel a llamará a esta función cada vez que se reciba una trama. La función para la notificación desde el nivel a tendrá como definición:

int Nombre_de_funcion (const uint8_t *Direccion_Remitente , int Tamano, const uint8_t * Mensaje, int Tipo, struct timeval * time);

Que es consistente con la declaración del tipo tpfNotificacionDesdeNivel a presentada anteriormente.

PARÁMETROS:

- const uint8_t • Direccion_Remitente: Puntero a la zona de memoria que contiene la dirección de la estación que envía la trama.
- int Tamano: Longitud en bytes del mensaje recibido (sólo los datos contenidos en el siguiente parámetro). Si se recibe O, quiere decir que no se ha recibido ninguna trama, y la llamada a la función se ha producido como resultado de la expiración del intervalo de timeout especificado en el último parámetro de la función lniciarNivel a.
- const uint8_t • Mensaje: Puntero a la zona de memoria que contiene el mensaje incluido en la trama recibida. En caso de haber recibido como tamaño O (llamada por time out), este puntero será NULL.
- int Tipo: Entero que indica el tipo de trama recibida (Ethertype). En caso de haber recibido como tamaño 0, éste valor será 0.
- int •struct timeval: estructura timeval que representa la marca temporal de la trama recibida. Esta marca temporal representa el tiempo Unix en el que que llega el paquete.

Al igual que en el caso de la función de notificación desde el nivel , esta función está pensada para realizar comprobaciones o chequeos en la trama sin modificarla, aceptando o denegando la validez de la trama recibida. En nuestro caso chequeará la trama y sólo aceptará las tramas correspondientes al canal u origen determinado.

Esta situación se indica a través del código de retorno:

VALOR DE RETORNO:

- OK (O): La trama se acepta.
- ERROR (-1): La trama debe rechazarse.

Puesto que retornar ERROR hace que el nivel a considere la trama como no válida y la descarte para su posterior proceso, este valor también podría utilizarse en caso de que la función de notificación desee realizar el proceso de almacenamiento de la trama por sí misma.