Prácticas de Redes de Ordenadores

5º curso de Ingeniería de Telecomunicación. Curso 2001-2002
Profesor: Daniel Morató Osés

Pamplona, 30 de septiembre de 2001

1 Librería para captura de paquetes

La captura de paquetes del nivel Ethernet en Linux requiere generalmente privilegios de superusuario. Para permitir a usuarios sin privilegios de administrador acceder a los paquetes que circulan por la red se ha diseñado un servidor que redirige las tramas Ethernet a aplicaciones de usuario. Para acceder al servicio se ha preparado una librería llamada *sockpromis*. El código objeto se encuentra en el fichero *libsockpromis.a* y las declaraciones en *sockpromis.h*, que se pueden encontrar en */opt3/ro/ficheros/sockpromis/*. A continuación se presentan las funciones exportadas por dicha librería.

1.1 int sp_new_sock(void)

Abre una nueva conexión con el servidor de la máquina local que se dedica a obtener las tramas. El valor que devuelve es el que se debe emplear en las demás funciones para hacer referencia a esta conexión. Devuelve -1 en caso de error.

1.2 int sp_recv(int s, void* buf, int len)

Esta función lee una nueva trama del nivel de enlace (con la cabecera pero sin la cola/trailer). El parámetro *s* debe ser el valor devuelto por la función *sp_new_sock()*. El puntero *buf* debe hacer referencia a una zona de memoria de al menos *len* bytes. Devuelve -1 en caso de error o de ser interrumpida, si no el número de bytes colocados en el buffer.

En paquetes de los protocolos TCP o UDP la sección de datos es sustituida por ceros por el servidor, manteniendo la longitud original del paquete.

1.3 Programa ejemplo

Fichero */opt3/ro/ficheros/sockpromis/ejemplos/ejemplo1.c*:

```
#include <stdio.h>
#include <sockpromis.h>

print_packet(char * buf, int tam)
{
   int i;

   for (i=0;(i<tam)&&(i<80);i++)
      printf("%x ", buf[i]&0xff);
   printf("\n");
}
```

```
main()
{
   int s;
   char buf[3000];
   int tam;

   s=sp_new_sock();
   if (s==-1) exit(-1);

   tam=sp_recv(s,buf,3000);
   if (tam==-1) exit(-1);

   print_packet(buf,tam);
}
```

Compilación:

```
[]% gcc /opt3/ro/ficheros/sockpromis/ejemplos/ejemplo1.c -I/opt3/ro/ficheros/sockpromis/include
 -L/opt3/ro/ficheros/sockpromis/lib -lsockpromis -o ejemplo_sockpromis
```

2 Práctica 1: Librería de decodificacion (3 ptos)

En esta práctica se debe crear una librería que exporte una función a la que se le pasa un paquete y llama a diferentes funciones según éste sea: Ethernet, ARP, IP, ICMP, UDP y/o TCP. Para eso se declaran unas variables que son punteros a las funciones que se van a llamar y a las que se les pasará el paquete desde el comienzo de la cabecera de ese nivel y el tamano de la PDU de ese nivel. Si no se le asigna valor a la variable función de un nivel no debe hacer nada si el paquete es de ese tipo.

El header de la librería, */opt3/ro/ficheros/easydecod/include/easydecod.h* :

```
extern int (*func_esETH)(char *pak, int tam);
extern int (*func_esARP)(char *pak, int tam);
extern int (*func_esIP)(char *pak, int tam);
extern int (*func_esICMP)(char *pak, int tam);
extern int (*func_esUDP)(char *pak, int tam);
extern int (*func_esTCP)(char *pak, int tam);

int easydecod(char *pak, int tam);
```

Se deja como ejemplo de la librería */opt3/ro/ficheros/easydecod/lib/libeasydecod.a* .

Formato de entrega: En el $HOME del grupo de prácticas se debe crear un directorio llamado *practica1* y en él dejar las fuentes y el makefile para obtener *libeasydecod.a* con solo hacer *make*.

3 Práctica 2: Programa de análisis de tráfico global (1 pto)

Crear un programa que calcule simples estadísticas de red. Este programa empleará la libreria *sockpromis* para la captura de paquetes y la de la práctica 1 haciendo que las funciones de cada protocolo cuenten paquetes, bytes, etc. Al menos debe dar la siguiente información: número de paquetes y bytes (en cada caso los bytes incluyen hasta la cabecera de ese protocolo) de los protocolos Ethernet, ARP, IP, ICMP, UDP y TCP. Opcionalmente puede dar otras estadísticas que se deseen: tráfico por servicio, por host, por tipo de error ICMP, etc.

Debe terminar al recibir un Ctrl-C y entonces sacar todas las estadísticas por la salida estándar.

El comando se llamará *analizared* y no necesitará opciones. Se deja como ejemplo */opt3/ro/ficheros/bin/analizared*

Formato de entrega: En el $HOME del grupo de prácticas se debe crear un directorio llamado *practica2* y en él dejar las fuentes y el makefile para obtener *analizared* con solo hacer *make*.

Funciones útiles: sigaction(2), signal(2)

4 Práctica 3: Programa de tráfico en tiempo real (2 ptos)

Crear un programa que saque tráfico en tiempo real:

trafico_tr [-p protocolo] [-i intervalo]

Donde *protocolo* puede ser: eth, arp, ip, udp o tcp. El parámetro *intervalo* es un entero que especifica el número de segundos en el cubo de muestreo. Los valores por defecto serán 'eth' y 1. [ ] significa que es un argumento opcional.

El programa saca por pantalla una línea con tres columnas cada *intervalo* segundos: el timestamp en segundos del comienzo del cubo (empezando en 0), el número de paquetes y el número de bytes de ese tipo en ese intervalo. Debe intentar sacar cada línea en cuanto concluya el intervalo.

Se deja como ejemplo */opt3/ro/ficheros/bin/trafico_tr*

Formato de entrega: En el $HOME del grupo de prácticas se debe crear un directorio llamado *practica3* y en él dejar las fuentes y el makefile para obtener *trafico_tr* con solo hacer *make*.

Funciones útiles: sigaction(2), signal(2), alarm(2)

5 Práctica 4: Túnel (4 ptos)

Disponemos de dos subredes conectadas a través del router R. Sin embargo, este router permite que circulen entre esas dos subredes solo los paquetes entre las maquinas A y B y siempre que los puertos origen y destino tengan el valor P, tanto TCP como UDP. Es decir, solo pueden transferirse entre esas subredes los paquetes UDP que cumplan que la dirección origen/destino sea la de la máquina A y el puerto origen/destino P y la dirección destino/origen la de B y el puerto destino/origen P o bien los paquetes TCP con iguales condiciones.

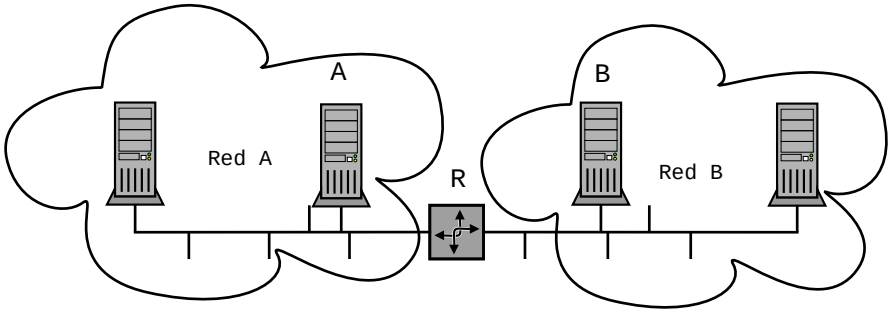

Figura 1: Redes con tráfico filtrado

Se desea poder emplear el servicio de telnet desde cualquier máquina de la subred A a cualquiera de la subred B. Para ello se debe crear, además de los programas que se coloquen en A y B para permitir que circule el servicio, un programa llamado *mtelnet <destino>* que permita acceder al servidor de telnet de las máquinas de la subred B.

Globalmente debería ser posible:

- Que haya un número indeterminado de conexiones en curso entre diferentes máquinas
- Que haya un número indeterminado de conexiones en curso entre las mismas máquinas

- Que el argumento de *mtelnet* sea un nombre o una dirección IP
- Que el nombre de la máquina como argumento de *mtelnet* no se pueda resolver en la subred A
- Que los flujos a través de R no se detengan porque se estén estableciendo nuevas conexiones de telnet
- Que se cumpla en lo posible el protocolo de telnet (empleando la aplicación telnet en el cliente y el servidor estándar en el destino)

En la maqueta la Red A corresponde a la red 10.1.5/24, la red B a la 10.1.6/24, la máquina A es pc1r5.maqueta.tlm.unavarra.es, la máquina B es pc1r6.maqueta.tlm.unavarra.es y el Router R es RouterC.

Para permitir las pruebas simultáneas de los diferentes grupos de prácticas los valores de puerto habilitados serán varios y cada grupo debe emplear el valor de P = 2100 + número_del_grupo.

Se recomienda emplear las máquinas de prácticas del laboratorio hasta tener terminada la práctica y recurrir a las máquinas pc1r5 y pc1r6 tan solo para comprobar al final que funciona correctamente.

Se ofrece una solución de ejemplo mediante los programas *tunel_int* y *tunel_ext* que están corriendo en las máquinas pc1r5 y pc1r6 respectivamente, que emplean entre ellos el puerto 2000 y la aplicación /usr/local/bin/mtelnet que se encuentra en la máquina pc2r5.maqueta.tlm.unavarra.es

Formato de entrega: En el $HOME del grupo de prácticas se debe crear un directorio llamado *practica4* y en él dejar las fuentes y el makefile para obtener con solo hacer *make* tanto *mtelnet* como las aplicaciones que se deban colocar en las máquinas A y B. Además se debe redactar una breve memoria (**NO** más de 5 páginas) explicando claramente la solución adoptada, el funcionamiento del protocolo a través de las máquinas A y B y con *mtelnet* y cómo poner en funcionamiento el sistema.

6 Actualizaciones

Actualizaciones, correcciones, comentarios y sugerencias respecto a las prácticas aparecerán en la página web de la assignatura http://www.tlm.unavarra.es/asignaturas/ro

Referencias

[1] Douglas E. Comer and David L. Stevens. *Internetworking with TCP/IP. Client-Server Programming and Applications; BSD Sockets Version*, volume 3. Prentice Hall.

[2] W. Richard Stevens. *Advanced programming in the UNIV environment.* Addison-Wesley, 1992.