12 Temporizador en la llamada bloqueante recvfrom

Elaborado por: Ukranio Coronilla

Una llamada a `recvfrom` se va a bloquear hasta que no llegue un mensaje que normalmente se ha solicitado mediante una llamada `sendto` previa. En ciertas aplicaciones conviene sacar al proceso del bloqueo después de cierto tiempo transcurrido donde no se ha recibido respuesta(timeout).

Para activar el temporizador se utiliza la función UNIX `setsockopt()` la cual permite manipular las opciones del socket. En este caso el tiempo se ajusta inicializando una estructura del tipo `timeval`, la cual contiene el miembro `tv_sec` para ajustar los segundos y `tv_usec` para los microsegundos. Por ejemplo para un tiempo de 2.5 segundos tendriamos:

```
#include <sys/time.h>

struct timeval tiempofuera;

tiempofuera.tv_sec = 2;
tiempofuera.tv_usec = 500000;
```

Hecha la inicialización de la variable `struct timeval` podemos utilizarla en la función `setsockopt` como sigue (véase man 7 socket):

```
setsockopt(s, SOL_SOCKET, SO_RCVTIMEO, (char *)&tiempofuera, sizeof(tiempofuera));
```

La variable global ***errno*** tomará el valor de la macro `EWOULDBLOCK` si el temporizador se ha activado. De este modo después de ejecutar la función `recvfrom` podremos imprimir un mensaje de que han pasado 2.5 segundos y no llegó ningún mensaje como sigue:

```
 #include <errno.h>
int n;

n = recvfrom(... )
if (n < 0) {
             if (errno == EWOULDBLOCK)
                 fprintf(stderr, "Tiempo para recepción transcurrido\n");
             else
                 fprintf(stderr, "Error en recvfrom\n");
}
```

Ejercicio 1: Añada una variable privada `struct timeval` y un método a la clase `SocketDatagrama` denominado:

```
setTimeout(time_t segundos, suseconds_t microsegundos);
```

para que un socket tenga asociado un temporizador. También se debe añadir una variable privada `bool timeout` cuyo valor será 1 si está activado el temporizador y 0 en caso contrario. Finalmente añada el método:

```
unsetTimeout( );
```

el cual deshabilitará el timeout dejando el socket en su estado inicial (véase man 7 socket).

Ejercicio 2: Añada un método `recibeTimeout` que realiza la misma función que el método recibe, pero además imprime el mensaje correspondiente si el tiempo de recepción ha transcurrido. Utilizando este método, elabore un cliente **monoproceso** que imprime todas las IP's de los servidores (descritos en el capítulo 9) que se encuentran activos en la subred.

Ejercicio 3: Además de imprimir el mensaje "Tiempo para recepción transcurrido", el método `recibeTimeout` deberá imprimir el tiempo que ha tardado desde que se ejecuta la función `recvfrom()` y hasta que se recibe el mensaje, si es que esto ha sucedido. Para ello utilizaremos la función `gettimeofday()` que inicializa su primer parámetro con el número de segundos más la fracción de microsegundos transcurridos desde el 1 de Enero de 1970 hasta este momento.

Un ejemplo de su uso se muestra a continuación:

```
#include <stdio.h>
#include <sys/time.h>

int main(void)
{
    struct timeval tiempo;

    gettimeofday(&tiempo, NULL);
    printf("segundos: %ld\n", tiempo.tv_sec);
    printf("Microsegundos: %ld\n", tiempo.tv_usec);

}
```

Utilice cualquiera de las funciones: `timeradd, timersub, timercmp, timerclear, timerisset`; y lleve a cabo la medición de tiempo que se solicita en este inciso.

Sugerencia: Para verificar si su programa es correcto mida el tiempo que tarda el siguiente par de instrucciones. Siempre deberá devolver muy aproximadamente el mismo valor, sin importar el número de ejecuciones.

```
sleep(1);
usleep(500000);
```