Capítulo 8. Código del Simulador

1 Makefile

```
CFLAGS=-D_POSIX_SOURCE -g
OBJ = sim.o worker.o protocol.o canal.o parametros.o
CC=gcc

all:    $(OBJ)
        $(CC)  -g -o sim $(OBJ)

clean:
        rm -f *.o *.bak sim

sim.o: common.h
worker.o:     common.h hilossec.h
protocol.o:   common.h hilossec.h protocol.h
canal.o: common.h hilossec.h
parametros.o: common.h
```

2 Common.h

```
/* En este fichero se encuentran las constantes y definiciones de tipos comunes a
 * todas los ficheros */

#include <stdio.h>

/* códigos de las respuestas enviadas por los nodos al main() */
#define OK      1               /* todo va bien */
#define NOTHING 2               /* el nodo no ha hecho nada */
#define END_FILE 3              /* se ha llegado al final del fichero a transmitir */

#define DELTA 10                /* debe ser mayor que el numero de temporizadores para que
                                        * cada temporizador pueda ir en un tick por separado, ne es
necesario
                                        * pero se mantiene por fidelidad al simulador de partida
                                        */

typedef unsigned long bigint;                   /* creacion del tipo bigint */

/* constantes generales */
#define TICK_SIZE (sizeof(bigint))
```

```
typedef struct {        /* almacena los fd necesarios para el hilo */
        int id;                 /* identifica al worker */
        int mr;                 /* lee del main */
        int mw;                 /* escribe al main */
        int wr;                 /* lee del otro nodo */
        int ww;                 /* escribe al otro nodo */
        int aux;                /* el nodo se escribe a sí mismo */
        int cr;                 /* para leer el estado del transmisor */
        int cw;                 /* escribe el tiempo al canal */
} filedesc;                     /* descriptores de fichero */

typedef struct {        /* parametros de un canal definidos por usuario */
        bigint t_prop;          /* tiempo de propagación (en ticks) */
        bigint rb;              /* regimen binario (en bits/seg) */
        bigint vp;              /* velocidad de propagacion (m/s) */
        bigint l;               /* longuitud (m) */
        int prob_error;         /* probabilidad de error en trama */
        int debug;              /* debug flags */
        int ticks_seg;          /* numero de tick que son un segundo */
} param_canal;

typedef struct {        /* parametros de un trabajador definidos por usuario */
        bigint timeout;         /* tiempo de expiración del temporizador de retranmisiones (en ticks)
*/
        bigint ack_timeout;     /* tiempo de expiración del temporizador de asentimientos (en ticks) */
        int debug_flags;        /* debug flags */
        int prob_error;         /* probabilidad de error en trama */
        int v_trx;              /* ventana de transmision */
        int v_rx;               /* ventana de recepción */
        int max_seq;            /* numero maximo de secuencia: ventana <= (max_seq+1)/2 */
                                        /* max_seq = 2^n-1 */
        int acks;               /* 1 si se usan tramas de asentimiento */
        int nacks;              /* 1 si se usan tramas de asentimiento negativo */
        FILE *enviar;           /* fichero a enviar */
        FILE *recibir;          /* fichero que recibir */
        bigint tickseg;         /* numero de ticks por segundo */
        bigint rb;              /* regimen binario */
        int mtu;                /* tamaño máximo de trama */
        unsigned semilla;       /* semilla para rand() */
} param_worker;

typedef struct {        /* parametros de la simulacion definidos por usuario */
        bigint t_total;         /* tiempo total de simulación (en ticks) */
        bigint ticks_seg;       /* numero de tick que son un segundo */
        bigint com_estad;       /* comienzo de las estadisticas (en ticks) */
        int acks;               /* 1 si se usan tramas de asentimiento */
        int nacks;              /* 1 si se usan tramas de asentimiento negativo */
        char fichero[255];      /* fichero donde almacenar los resultados */
        unsigned semilla;       /* semilla para rand() */
} param_sim;
```

```
typedef struct {       /* Parametros de la simulacion */
        param_canal canal0;             /* parametros del canal 0 (0 ->1) */
        param_canal canal1;             /* parametros del canal 1 (1 ->0) */
        param_worker worker0;   /* parametros del trabajador 0 */
        param_worker worker1;   /* parametros del trabajador 1 */
        param_sim simulacion;   /* parametros globales de la simulacion */
} arguments;
```

3 Hilossec.h

```
/* Aquí se encuentran las constantes y definiciones de tipos comunes a los ficheros
 * worker.c, parametros.c y protocol.c */

/* máscaras de traza */
#define TRANSMITS           0x0001          /* transmisión de trama */
#define DELIVERS    0x0002          /* paquete entregado */
#define SENDS       0x0004    /* trama pasa al nivel fisico */
#define RECEIVES    0x0008          /* trama recibida */
#define TIMEOUTS    0x0010          /* expira un temporizador */
#define PERIODIC    0x0020 /* traza pariódica para uso con simulaciones largas */

#define MAX_PKT 4           /* determina el tamaño de la cabecera de la trama en bytes */
#define MAX_LINEA 100     /* determina el máximo tamaño de línea que va en una trama */

#define MAX_QUEUE 5             /* numero máximo de tramas en el buffer */
#define NO_EVENT -1                             /* no se produce ningún evento */

typedef unsigned int seq_nr;                    /* numero de secuencia o asentimiento */

typedef enum {false, true} boolean;             /* creacion del tipo boolean */

/* tipos de eventos posibles */
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;

typedef struct {                                        /* definicion de la información de una trama */
        unsigned char data[MAX_PKT];            /* numero de secuencia de nivel de red */
        unsigned char linea[MAX_LINEA];         /* linea del fichero que se transmite */
} packet;

typedef enum {data, ack, nak, end} frame_kind;      /* definicion del tipo de trama */

typedef enum {periodic, tmout, ack_tmout, send_pkt, del_pkt, del_ini,
        good_rec, bad_rec, trx} traza_kind;             /* definicion del tipo de traza */

typedef struct {        /* descripcion de las tramas transmitidas por esta capa */
  frame_kind kind;      /* tipo de trama */
  seq_nr seq;           /* numero de secuencia */
  seq_nr ack;           /* numero de asentimiento */
  packet info;   /* paquete a entregar al nivel de red */
  bigint t_trans;       /* en que tick fue transmitida por primera vez */
} frame;

#define FRAME_SIZE (sizeof(frame))          /* tamano de una trama de datos */

struct lista {    /* nodo de la lista doblemente enlazada del canal */
        frame fr;                       /* trama */
        bigint timer;           /* almacena el tick en que la trama cambia de estado */
        struct lista *ant;      /* puntero al nodo anterior */
        struct lista *sig;      /* puntero al nodo siguiente */
};

typedef struct lista nodo;      /* definicion del tipo de datos nodo */

typedef struct {        /* estadisticas */
        int data_sent;                  /* numero de tramas enviadas */
        int data_retransmitted;         /* numero de tramas retrasnmitidas */
        int good_data_recd;             /* numero de tramas recibidas sin error */
        int cksum_data_recd;  /* numero de tramas recibidas con error */
        int acks_sent;                  /* numero de asentimiento enviados */
        int good_acks_recd;             /* numero de asentimiento recibidos sin error */
        int cksum_acks_recd;  /* numero de asentimiento recibidos con error */
        int payloads_accepted;          /* numero de paquetes pasados al nivel de red */
        int timeouts;                   /* numero de timeouts */
        int ack_timeouts;               /* numero de timeouts de asentimiento */
        bigint delay;                   /* retardo de las tramas */
        int delay_counted;              /* numero de los paquetes de los que se ha contado el retardo */
        bigint round_delay;             /* retardo de vuelta de las tramas */
        int trip_cuonted;               /* numero de las tramas de las que se cuenta el retardo de vuelta */
        float uso;                              /* uso de la línea en ticks */
        bigint ttotal;                  /* tiempo total de simulacion */
        bigint bytes_trx;               /* numero de bits transmitidos */
        bigint rb;                              /* regimen binario */
        bigint tickseg;                 /* numero de ticks por segundo */
} statistics;

typedef struct {        /* todas las variables relacionadas con los temporizadores */
        bigint *ack_timer;                      /* puntero a la tabla de los temporizadores;
                                                                cada temporizador almacena el tick en el que
                                                                debe expirar */
        int nr_timers;                          /* numero de temporizadores */
        unsigned int *seqs;                     /* puntero a la tabla con los numeros de secuencia
                                                                * de la trama correspondiente a cada temporizador
*/
        seq_nr oldest_frame;            /* indica cual trama es la que expiro su temporizador */
        bigint lowest_timer;            /* valor del temporizador mas bajo */
        bigint aux_timer;                       /* temporizador de asentimiento */
        int offset;                             /* para prevenir que varios temporizadores expiren
                                                                * en el mismo tick */
        bigint tick;                            /* tick actual */
        event_type event;                       /* ultimo evento */
} timers;

typedef struct {        /* buffer donde se almacenan las tramas que llegan para su uso posterior */
        frame *queue;                   /* puntero a la trama que hace de buffer */
        int in;                                 /* donde poner la siguiente trama */
        int out;                                /* de donde sacar la siguiente trama */
        int nframes;                    /* numero de tramas almacenadas */
        frame last_frame;               /* trama que acaba de ser recibida */
        nodo *list;                             /* puntero a nodo */
        seq_nr frm_exp;                 /* numero de secuecia de la trama que se espera */
        int max_seq;                    /* numero máximo de secuencia */
```

```
        boolean trans;                  /* indica si el transmisor esta listo */
} buffer;
```

4 Protocol.h

```
/* Fichero de cabecera de protocol.c */
#include "hilossec.h"

/* Las siguientes funciones están implementadas en el fichero worker.c */

/* Espera a que ocurra algun evento y devuelve el tipo de evento */
event_type wait_for_event(filedesc fd, timers *t, int n_l_s, buffer *b,
                                        boolean *fin_fich_rec, param_worker args, statistics *st);

/* Recoge un paquete del nivel de red */
packet from_network_layer(unsigned int *next_net_pkt, FILE *fich, boolean *eof, int mtu);

/* Entrega un paquete al nivel de red */
unsigned int to_network_layer(packet p, unsigned int last_pkt_given, int id,
                                                bigint tick, param_worker args, statistics *st);

/* recoge una trama del nivel fisico y la almacena en r */
void from_physical_layer(frame *r, buffer b);

/* Pasa una trama al nivel fisico para que la transmita */
nodo *to_physical_layer(frame s, nodo *list, int id, bigint tick, param_worker args,
                                        statistics *st);

/* Arranca un temporizador de retransmision y habilita el evento timeout */
void start_timer(seq_nr k, bigint timeout_interval, timers *t);

/* Para un temporizador de retransmision */
void stop_timer(seq_nr k, timers *t);

/* Arranca el temporizador de asentimiento y habilita el evento ack_timeout */
void start_ack_timer(timers *t, bigint ack_timeout);

/* Para el temporizador de asentimiento y deshabilita el evento ack_timeout */
void stop_ack_timer(bigint *aux_timer);

/* Permite al nivel de red que genere un evento network_layer_ready */
int enable_network_layer(void);

/* Prohibe al nivel de red que genere un evento network_layer_ready */
int disable_network_layer(void);

/* Inicializa las variables estadísticas */
statistics inicia_estadisticas(void);

/* Arranca el protocolo a utilizar */
void protocol(filedesc fd, param_worker args);
```

5 Sim.c

```
/* Es este fichero se encuentra la función principal del simulador, main(). Se le pasan dos
 * argumentos para leer la línea de comando introducida por el usuario, donde éste debe
 * introducir el nombre del fichero de entrada en el que se recogen los parámetros de la
 * simulación.
 */

/* El programa consta de cinco hilos. El primero es el de la función main(). Otros dos hilos
 * son dedicados a las funciones de los nodos y otros dos a los dos canales. Cada hilo se
 * ejecuta "independientemente". Para comunicarse entre sí los hilos utilizan pipes. Mediante
 * los pipes se intercambia el tiempo de la simulación, las tramas o paquetes y otra
 * información estadística y de control.
 */

/* En la versión original cada tick lo ejecutaba un nodo, ahora los dos nodos y los dos canales
 * reciben el tiempo cada tick, además pueden producirse varios eventos en el mismo tick
 */

/* Simulator for the protocols in chapter 3 of
 *    "Computer Networks, 3rd ed. by Andrew S. Tanenbaum.
 * SIM mejorado 1.0 ultimas mejoras: 15/5/02
 *
 *                  -> Las tramas Ack y Nack que llegan con error no producen Nacks
 *                  -> Se manda un Nack siempre que llega una trama de datos con error
 */

#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <unistd.h>
#include <time.h>
#include "common.h"

#define DEADLOCK (3 * args.worker0.timeout)       /* define que es un bucle infinito */
#define MANY 256                /* suficiente para vaciar el pipe al final */

typedef struct {          /* estructura donde almacenar los pipes */
        int pipe01[2];
        int pipe02[2];
        int pipe03[2];
        int pipe04[2];
        int pipe05[2];
        int pipe06[2];
        int pipe07[2];
        int pipe08[2];
        int pipe09[2];
        int pipe10[2];
        int pipe11[2];
        int pipe12[2];
} pipes;
```

```
/* Prototipos */
int main(int argc, char *argv[]);
arguments parse_args(int argc, char *argv[]);
void set_up_pipes(pipes *p);
void fork_off_workers(pipes p, arguments args);
void terminate(float dur, bigint time, char *s, pipes p, arguments args);
void protocol(filedesc fd, param_worker args);
void canal(filedesc fd, param_canal);

int main(int argc, char *argv[])
/* Esta función implementa el hilo principal del programa. En esta función se lleva el tiempo
 * de simulación y se transmite mediate los pipes a los otros hilos. Además se controla cuando
 * termina el tiempo de simulación, se monitorizan los nodos y se manda la señal de fin de
 * simulación cuando ambos nodos se han quedado sin nada que transmitir.
 */
{
        bigint tick = 0;                /* tiempo actual de la simulación */
        int process = 0;                /* de que proceso es el turno */
        int rwfd[2], wwfd[2];   /* descriptores de fichero para hablar con los nodos */
        bigint word;            /* almacena los mensajes para los nodos */
        int hanging[2];                 /* numero de tiempos que un nodo está inactivo */
        arguments args;                 /* estructura donde se guardan los parámetros de la simulación */
        pipes p;                        /* estructura donde se almacenan los pipes para la comunicación */
        int fin_ficheros = 0;   /* cuando esta variable alcanza valor 2 ambos nodos se han quedado
                                                sin nada que transmitir */
        float duracion;                 /* duración en tiempo real de la simulación */
        int i;                          /* variable índice */

        duracion = clock();     /* almacena el tiempo real cuando comienza la simulación */
        setvbuf(stdout, (char *) 0, _IONBF, (size_t) 0);        /* deshabilita el buffering */
        args = parse_args(argc, argv);          /* analiza el fichero de entrada y lee los parámetros */
        set_up_pipes(&p);                               /* crea los pipes */
        fork_off_workers(p, args);              /* fork off the worker processes */
        srand(args.simulacion.semilla);         /* establece la semilla para la función rand() */

        /* bucle principal de la simulación, se ejecuta hasta que se alcanza el tiempo máximo de
           simulación o los dos nodos se quedan sin nada que transmitir. En cada vuelta se manda
           a los dos nodos el tiempo de simulación, a la vez que se sincronizan y se recoge su
           estado */
        while (tick < args.simulacion.t_total) {
                process = rand() & 1;           /* elige un proceso para arrancar primero*/
                tick = tick + DELTA;            /* aumenta el tiempo de simulación */
                rwfd[0] = (process == 0 ? p.pipe04[0] : p.pipe06[0]);           /* selecciona 1er nodo */
                rwfd[1] = (process == 1 ? p.pipe04[0] : p.pipe06[0]);           /* y establece los fd */
                wwfd[0] = (process == 0 ? p.pipe03[1] : p.pipe05[1]);
                wwfd[1] = (process == 1 ? p.pipe03[1] : p.pipe05[1]);

                /* hace lo mismo para los dos nodos, pero a uno le da la orden de continuar primero */
                for(i = 0; i <= 1; i++){
                        /* lee las respuestas, primero del canal y depues del nodo */
                        if (read(rwfd[i], &word, TICK_SIZE) != TICK_SIZE) terminate(0, tick, "", p, args);

                        /* procesa la respuesta */
                        if (word == OK) hanging[process ^ i] = 0;       /* todo va bien */
                        if (word == NOTHING) hanging[process ^ i] += DELTA;             /* nodo inactivo */
                        if (word == END_FILE) fin_ficheros++;                   /* no tiene nada que transmitir */
                        /* comprueba fin de los dos ficheros */
                        if(fin_ficheros == 2) {         /* lo dos nodos se han quedado sin nada que transmitir */
                                /* establece el tiempo total de simalción al siguiente tick para que finalice
                                   la simulación inmediatamente */
                                args.simulacion.t_total = tick + 1 * DELTA;
                                fin_ficheros++;                 /* para que no pueda darse la situación de nuevo */
                        }
                        /* comprueba si hay bucle infinito, para ello uno de los nodos debe permanecer
                           demasiado tiempo inactivo */
                        if (hanging[0] >= DEADLOCK && hanging[1] >= DEADLOCK)
                                terminate(0, tick, "A deadlock has been detected", p, args);

                        /* manda el tiempo al nodo seleccionado para permitirle continuar */
                        if (write(wwfd[i], &tick, TICK_SIZE) != TICK_SIZE)
                                terminate(0, tick, "Main could not write to worker", p, args);
                }
        }
        /* Cierra todos los ficheros */
        fclose(args.worker0.enviar);
        fclose(args.worker0.recibir);
        fclose(args.worker1.enviar);
        fclose(args.worker1.recibir);
        /* calcula el tiempo real que ha durado la simulación */
        duracion = clock() - duracion;
        duracion = (1000 * duracion) / (float) CLOCKS_PER_SEC;
        /* la simulación ha llegado a su fin */
        terminate(duracion, tick, "Fin de la simulación", p, args);
        return;
}

void set_up_pipes(pipes *p)
/* Crea doce pipes para que el main pueda comunicarse con los nodos y estos con los canales
 * N0 = nodo 0; N1 = nodo 1; C0 = canal 0; C1 = canal 1; M = main
 * en al guía de usuario aparece un gráfico para mejor comprensión
 */
{
 int fd[2];

 pipe(p->pipe01);       /* para tramas de N0 a C0 */
 pipe(p->pipe02);       /* para tramas de N1 a C1 */
 pipe(p->pipe03);       /* tiempo de M a N0 */
 pipe(p->pipe04);       /* estado de N0 a M */
 pipe(p->pipe05);       /* tiempo de M a N1 */
 pipe(p->pipe06);       /* estado de N0 a M */
 pipe(p->pipe07);       /* para tramas de C0 a N1 */
 pipe(p->pipe08);       /* C1 a N1 para estado del transmisor */
 pipe(p->pipe09);       /* para tramas de C1 a N0 */
 pipe(p->pipe10);       /* C0 a N0 para estado del transmisor */
 pipe(p->pipe11);       /* tiempo de N1 a C1 */
 pipe(p->pipe12);       /* C1 a N1 para estado del transmisor */

 return;
}

void fork_off_workers(pipes p, arguments args)
/* Crea los hilos de los nodos y de los canales, además cierra los extremos de los pipes
 * que no son necesarios en cada hilo, y asigna los descriptores de fichero a los extremos
 * de los pipes que corresponde
```

```
 */
{
	filedesc fd;		/* estructura con los descriptores de fichero */
	/* las siguientes estructuras solo son necesarias para que al cerrar un extremo de un
	   pipe y producirse la señal correspondiente no finalice la ejecución del programa
	   por un supuesto error */
	struct sigaction act, oact;
	act.sa_handler = SIG_IGN;

	/* este es el hilo padre, que luego será el main, pero antes ha de bifurcarse en los
	   hilos de los nodos y los canales */
	/* en la siguiente intrucción hay que poner un == tras el fork() para que sea el hilo
	   del main el hilo principal del programa. Se puede poner !=, y es interesante para
	   el debug pero entonces el windows detecta el final del programa antes de que termine
	   el hilo del main y se cierre el fichero de salida, lo que puede producir un error con
	   InGraSE */
	if (fork() == 0) {
		/* crea los hilos de los canales y nodos */
		if (fork() == 0) {
			/* estos son los nodos */
			if (fork() != 0) {
				/* este es el codigo del nodo 1, arranca el protocolo */
				/* evita que la señal de pipe roto interrumpa la ejecución del programa */
				sigaction(SIGPIPE, &act, &oact);
				setvbuf(stdout, (char *)0, _IONBF, (size_t)0);		/* deshabilita el buffering */
				close(p.pipe01[0]);		/* cierra los pipes que no utiliza */
				close(p.pipe01[1]);
				close(p.pipe02[0]);
				close(p.pipe03[0]);
				close(p.pipe03[1]);
				close(p.pipe04[0]);
				close(p.pipe04[1]);
				close(p.pipe05[1]);
				close(p.pipe06[0]);
				close(p.pipe08[0]);
				close(p.pipe08[1]);
				close(p.pipe09[0]);
				close(p.pipe09[1]);
				close(p.pipe10[0]);
				close(p.pipe10[1]);
				close(p.pipe11[0]);
				close(p.pipe12[1]);

				fd.id = 1;				/* identificador 1 */
				fd.mr = p.pipe05[0];	/* para leer el tick del main */
				fd.mw = p.pipe06[1];	/* para escribir la respuesta al main */
				fd.wr = p.pipe07[0];	/* para recibir tramas del otro */
				fd.ww = p.pipe02[1];	/* para mandar tramas al otro nodo */
				fd.aux = p.pipe07[1];	/* para escribir la bandera */
				fd.cr  = p.pipe12[0];	/* para leer el estado del transmisor */
				fd.cw = p.pipe11[1];	/* para mandar el tick al canal */

				protocol(fd, args.worker1);	/* inicia el protocolo */
				/* la siguiente función no debe ejecutarse nunca */
				terminate(0, 0, "Imposible. Protocolo terminado", p, args);
			} else {
				/* este es el codigo del nodo 0, arranca el protocolo */
				/* evita que la señal de pipe roto interrumpa la ejecución del programa */
				sigaction(SIGPIPE, &act, &oact);
				setvbuf(stdout, (char *)0, _IONBF, (size_t)0);		/* deshabilita el buffering */
				close(p.pipe01[0]);		/* cierra los pipes que no utiliza */
				close(p.pipe02[0]);
				close(p.pipe02[1]);
				close(p.pipe03[1]);
				close(p.pipe04[0]);
				close(p.pipe05[0]);
				close(p.pipe05[1]);
				close(p.pipe06[0]);
				close(p.pipe06[1]);
				close(p.pipe07[0]);
				close(p.pipe07[1]);
				close(p.pipe09[0]);
				close(p.pipe10[1]);
				close(p.pipe11[0]);
				close(p.pipe11[1]);
				close(p.pipe12[0]);
				close(p.pipe12[1]);

				fd.id = 0;				/* identificador 0 */
				fd.mr = p.pipe03[0];	/* para leer el tick del main */
				fd.mw = p.pipe04[1];	/* para escribir la respuesta al main */
				fd.wr = p.pipe08[0];	/* para recibir tramas del otro */
				fd.ww = p.pipe01[1];	/* para mandar tramas al otro nodo */
				fd.aux = p.pipe08[1];	/* para escribir la bandera */
				fd.cr = p.pipe10[0];	/* para leer el estado del transmisor */
				fd.cw = p.pipe09[1];	/* para mandar el tick al canal */

				protocol(fd, args.worker0);	/* inicia el protocolo */
				/* la siguiente función no debe ejecutarse nunca */
				terminate(0, 0, "Imposible. Protocolo terminado", p, args);
			}
		} else {
			/* estos son los canales */
			if (fork() == 0) {
				/* este es el codigo del canal 1, arranca el protocolo */
				/* evita que la señal de pipe roto interrumpa la ejecución del programa */
				sigaction(SIGPIPE, &act, &oact);
				setvbuf(stdout, (char *)0, _IONBF, (size_t)0);		/* deshabilita el buffering */
				close(p.pipe01[0]);		/* cierra los pipes que no utiliza */
				close(p.pipe01[1]);
				close(p.pipe03[0]);
				close(p.pipe03[1]);
				close(p.pipe04[0]);
				close(p.pipe04[1]);
				close(p.pipe05[0]);
				close(p.pipe05[1]);
				close(p.pipe06[0]);
				close(p.pipe06[1]);
				close(p.pipe07[0]);
				close(p.pipe07[1]);
				close(p.pipe08[0]);
				close(p.pipe09[0]);
				close(p.pipe09[1]);
				close(p.pipe10[0]);
```

```
                close(p.pipe10[1]);
                close(p.pipe11[1]);
                close(p.pipe12[0]);

                fd.id = 1;                              /* identificador 1 */
                fd.mr = p.pipe11[0];    /* para leer el tick del nodo */
                fd.mw = p.pipe12[1];    /* para escribir el estado del transmisor */
                fd.wr = p.pipe02[0];    /* para recibir tramas del nodo 1 */
                fd.ww = p.pipe08[1];    /* para enviar tramas al nodo 0 */
                fd.aux = p.pipe02[1];   /* para ecribir la bandera */

                canal(fd, args.canal1);         /* inicia el canal */
                /* la siguiente función no debe ejecutarse nunca */
                terminate(0, 0, "Imposible. Canal terminado", p, args);
        } else {
                /* este es el codigo del canal 0, arranca el protocolo */
                /* evita que la señal de pipe roto interrumpa la ejecución del programa */
                sigaction(SIGPIPE, &act, &oact);
                setvbuf(stdout, (char *)0, _IONBF, (size_t)0);          /* deshabilita el buffering */
                close(p.pipe02[0]);             /* cierra los pipes que no utiliza */
                close(p.pipe02[1]);
                close(p.pipe03[0]);
                close(p.pipe03[1]);
                close(p.pipe04[0]);
                close(p.pipe04[1]);
                close(p.pipe05[0]);
                close(p.pipe05[1]);
                close(p.pipe06[0]);
                close(p.pipe06[1]);
                close(p.pipe07[0]);
                close(p.pipe08[0]);
                close(p.pipe08[1]);
                close(p.pipe09[1]);
                close(p.pipe10[0]);
                close(p.pipe11[0]);
                close(p.pipe11[1]);
                close(p.pipe12[0]);
                close(p.pipe12[1]);

                fd.id = 0;                              /* identificador 0 */
                fd.mr = p.pipe09[0];    /* para leer el tick del nodo */
                fd.mw = p.pipe10[1];    /* para escribir el estado del transmisor */
                fd.wr = p.pipe01[0];    /* para recibir tramas del nodo 0 */
                fd.ww = p.pipe07[1];    /* para enviar tramas al nodo 1 */
                fd.aux = p.pipe01[1];   /* para ecribir la bandera */

                canal(fd, args.canal0);         /* inicia el canal */
                /* la siguiente función no debe ejecutarse nunca */
                terminate(0, 0, "Imposible. Canal terminado", p, args);
        }
    }
} else {
    /* este es el hilo del main */
    /* evita que la señal de pipe roto interrumpa la ejecución del programa */
    sigaction(SIGPIPE, &act, &oact);
    setvbuf(stdout, (char *)0, _IONBF, (size_t)0);          /* deshabilita el buffering */
    close(p.pipe01[0]);             /* cierra los pipes que no utiliza */
                close(p.pipe01[1]);
                close(p.pipe02[0]);
                close(p.pipe02[1]);
                close(p.pipe03[0]);
                close(p.pipe04[1]);
                close(p.pipe05[0]);
                close(p.pipe06[1]);
                close(p.pipe07[0]);
                close(p.pipe07[1]);
                close(p.pipe08[0]);
                close(p.pipe08[1]);
                close(p.pipe09[0]);
                close(p.pipe10[1]);
                close(p.pipe11[0]);
                close(p.pipe12[1]);
                return;
        }
        /* la siguiente función no debe ejecutarse nunca */
        terminate(0, 0, "Imposible. Error en fork_off_workers().Programa terminado", p, args);
}

void terminate(float dur, bigint time, char *s, pipes p, arguments args)
/* Finaliza la simulación enviando a cada nodo la señal de fin. Además lee de cada nodo
 * los datos estadisticos necesarios para calcaular algunos resultados y les da tiempo
 * a escribir sus datos en el fichero de salida
 */
/* ya no es necesario leer datos de los nodos y canales pues estos escriben todos sus datos
 * pero se conserva el codigo por si vale para uso futuro
 */
{
        int n, k1, k2;                          /* variables indice */
        int res1[MANY], res2[MANY];     /* almacena los datos leidos de los nodos */
        int acc, sent;                          /* resultados de los calculos */
        float eff, leff, ttime;         /* para realizar los calculos */
        bigint zero = 0;                        /* señal de fin de simulación */

        /* inicializa las variables a 0 */
        for (n = 0; n < MANY; n++) {res1[n] = 0; res2[n] = 0;}
        /* Manda la senal de fin de simulacion y espera a que escriban en el fichero de salida */
        write(p.pipe03[1], &zero, TICK_SIZE);           /* a M0 */
        sleep(1);
        write(p.pipe05[1], &zero, TICK_SIZE);           /* a M1 */
        sleep(2);

        /* limpia el pipe, y lee los datos del nodo */
        n = read(p.pipe04[0], res1, MANY*sizeof(int));
        k1 = 0;
        while (res1[k1] != 0) k1++;
        k1++;                           /* res1[k1] = tramas aceptadas, res1[k1+1] = tramas enviadas */

        /* limpia el pipe, y lee los datos del nodo */
        n = read(p.pipe06[0], res2, MANY*sizeof(int));
        k2 = 0;
        while (res2[k2] != 0) k2++;
        k2++;                           /* res2[k1] = tramas aceptadas, res2[k1+1] = tramas enviadas */

        if (strlen(s) > 0) {
```

```
                /* hace las sumas pertinentes */
                acc = res1[k1] + res2[k2];
                sent = res1[k1+1] + res2[k2+1];
                printf("\nGeneral :\n");
                if (sent > 0) {
                        /* realiza los cálculos de algunos parametros que ya no hacen falta */
                        eff = ((float)acc/(float)sent) * 100;
                        // printf("Efficiency (payloads accepted/data pkts sent) = %.2f %%\n",eff);
                        leff = 10 * (float) res1[k1+1] * (float) DELTA * (float) 5;
                        leff /= (float) time;
                        // printf("Uso (t. ocupada/t. total) de la línea 0->1 = %.2f %%\n", leff);
                        leff = 10 * (float) res2[k2+1] * (float) DELTA * (float) 5;
                        leff /= (float) time;
                        // printf("Uso (t. ocupada/t. total) de la línea 1->0 = %.2f %%\n", leff);
                }
                /* calcula el tiempo real que duró la simulación */
                time /= DELTA;
                ttime = (float) (time) / (float) args.simulacion.ticks_seg;
                ttime *= 1000;
                /* escribe los datos necesarios al final del fichero de salida*/
                printf("Duración de la simulación = %.2f ms.\n", dur);
                printf("Tiempo total simulado = %.2f ms.\n%s.\n", ttime, s);
        }
        close(fileno(stdout));   /* cierra el fichero de salida */
        exit(0);
}
```

6 Parametros.c

```
/* En este fichero se recogen las funciones que se utilizan para procesar el fichero
 * de entrada del simlulador. Este fichero debe terner una estructura fija que se irá
 * explicando conforme se describan las distintas funciones que lo leen. Tambien hay una
 * descripción de él en el manual de usuario
 */

/* La función principal es parse_args() que devuelve una estructura con todos los parámetros
 * contenidos en el fichero. Esta función es llamada por main() y se ejecuta al principio
 * de cada simulación
 */

#include "common.h"

/* prototipos de las funciones */
int iguales(char *uno, char *dos);
param_sim parse_sim(FILE *fich);
param_worker parse_nodo(FILE *fich);
param_canal parse_canal(FILE *fich);
FILE *avanza(FILE *fich, char *buf);
int calcula_max_seq(int v1, int v2, int v3, int v4);
int exp2(int i);

arguments parse_args(int argc, char *argv[])
/* Procesa el fichero de entrada del simulador. Recive como parámetros los introducidos
 * por el usuario en la línea de comandos: argc y argv[], y devuelve una estructura
 * con los parámetros
 */
{
        FILE *fich;             /* puntero al fichero de entrada */
        char buf[100];  /* buffer para la lectura desde el fichero */
        char desc[100];         /* descripción de la simulación */
        int fd, fout;   /* descriptores de fichero para el fichero de salida y stdout */
        int aux;                /* variable auxiliar */
        arguments args;         /* estructura donde almacenar los parámetros */
        param_sim sim;          /* estructura para almacenar los parámetros generales */
        param_worker nodo0, nodo1;  /* estructuras para almacenar los parámetros de nodo */
        param_canal canal0, canal1;   /* estructuras para almacenar los parámetros de canal */

        if(argc != 2) {         /* comprueba que hay argumento de fichero de entrada */
                printf("Se requiere como parámetro un fichero de entrada.\n");
                exit(1);
        }

        fich = fopen(argv[1], "rt");            /* abre fichero entrada */
        if(fich == NULL) {
                printf("Error al abrir fichero\n");
                exit(1);
        }

        fscanf(fich, "%[^\n]", desc);           /* quita primera línea y demás comentarios */
        fich = avanza(fich, buf);

        if(!iguales(buf, "PARAMETROS")) {       /* comprueba que ponga PARAMETROS */
                printf("ERROR: %s\n", buf);     /* al comenzar */
                exit(1);
        }

        fich = avanza(fich, buf);               /* salta comentarios y lee siguente palabra */

        do {                                                    /* lee el fichero de entrada */
                if(iguales(buf, "SIMULACION")) {
                        sim = parse_sim(fich);                  /* lee parametros de simulacion */
                } else if(iguales(buf, "NODO")) {
                        fscanf(fich, "%d", &aux);               /* lee parametros de nodo */
                        /* y segun el indice los guarda en una estructura o en otra */
                        (aux == 0 ? nodo0 : nodo1) = parse_nodo(fich);
                        if(aux != 0 && aux != 1) printf("ERROR: %d", aux);
                } else if(iguales(buf, "CANAL")) {
                        fscanf(fich, "%d", &aux);               /* lee parametros de canal */
                        /* y segun el indice los guarda en una estructura o en otra */
                        (aux == 0 ? canal0 : canal1) = parse_canal(fich);
                        if(aux != 0 && aux != 1) printf("ERROR: %d", aux);
                } else {
                        /* si no se ha leído ninguna de esas palabras es que ha habido un error
                           de lectura o en la estructura del fichero */
                        printf("Error en fichero de entrada: %s\n", buf);
                        exit(1);
                }
                /* avanza hasta la siguiente palabra saltandose los comentarios */
                fich = avanza(fich, buf);
        /* la cadena "/PARAMETROS" señaliza el final del fichero */
        } while(!iguales(buf, "/PARAMETROS"));          /* fin de fichero */
```

```
/* cierra el fichero */
fclose(fich);

/* Sustituye la salida estandar por el fichero de salida */
fich = fopen(sim.fichero, "wt");
if (fich == NULL) {                /* problemas al abrir el fichero */
        printf("No se puede abrir el fichero %s\n", sim.fichero);
        exit(1);
}
fd = fileno(fich);
fout = fileno(stdout);
if(dup2(fd,fout) != fout) {                /* error al desviar la salida estandar */
        printf("Error al desviar la salida estandar");
        exit(1);
}

/* calcula el maximo numero de secuencia */
nodo0.max_seq = calcula_max_seq(nodo0.v_rx, nodo1.v_rx, nodo0.v_trx, nodo1.v_trx);
nodo1.max_seq = nodo0.max_seq;

/* Presenta los datos del fichero de entrada por pantalla */
printf("Sim v1.0. Versión creada por Juan Luis Millán Romera.\n");
printf("%s\nParámetros de la SIMULACION:\n", desc);
printf("Fichero de salida:\t\t%s\n", sim.fichero);
printf("Tiempo total de simulación:\t%d\t\t", sim.t_total);
printf("Ticks por segundo:\t%d\n", sim.ticks_seg);
printf("Máximo número de secuencia:\t%d\t\t", nodo0.max_seq);
printf("Semilla:\t\t%u\n", sim.semilla);
printf("NODO O:\n");
printf("Temp:\t%d\tTemp_ack:\t%d\tTraza:\t%d\n",
        nodo0.timeout, nodo0.ack_timeout, nodo0.debug_flags);
printf("Vent. trx:\t%d\tVent. rx:\t%d\tMtu:\t%d\n",
        nodo0.v_trx, nodo0.v_rx, nodo0.mtu);
printf("NODO 1:\n");
printf("Temp:\t%d\tTemp_ack:\t%d\tTraza:\t%d\n", nodo1.timeout,
        nodo1.ack_timeout, nodo1.debug_flags);
printf("Vent. trx:\t%d\tVent. rx:\t%d\tMtu:\t%d\n",     nodo1.v_trx, nodo1.v_rx,
        nodo1.mtu);
printf("CANAL 0:\n");
printf("Reg. bin.:\t%d\tVel. Prop.:\t%d\t", canal0.rb, canal0.vp);
printf("Long.:\t%d\tError:\t%d\n", canal0.l, canal0.prob_error);
printf("CANAL 1:\n");
printf("Reg. bin.:\t%d\tVel. Prop.:\t%d\t", canal1.rb, canal1.vp);
printf("Long.:\t%d\tError:\t%d\n", canal1.l, canal1.prob_error);
printf("Comienza la simulación.\n");

/* calculos necesarios */
/* tiempo de propagacion */
canal0.t_prop = (canal0.l * sim.ticks_seg) / canal0.vp;
canal1.t_prop = (canal1.l * sim.ticks_seg) / canal1.vp;
/* temporizadores */
nodo0.timeout = (nodo0.timeout * sim.ticks_seg) / 1000;
nodo1.timeout = (nodo1.timeout * sim.ticks_seg) / 1000;
nodo0.ack_timeout = (nodo0.ack_timeout * sim.ticks_seg) /1000;
nodo1.ack_timeout = (nodo1.ack_timeout * sim.ticks_seg) /1000;
/* duracion simulacion */
        sim.t_total *= sim.ticks_seg;
        /* probabilidad de error de trama */
        nodo0.prob_error = canal1.prob_error;
        nodo1.prob_error = canal0.prob_error;
        /* banderas de traza */
        canal0.debug = nodo0.debug_flags;
        canal1.debug = nodo1.debug_flags;
        /* ticks por segundo */
        canal0.ticks_seg = sim.ticks_seg;
        canal1.ticks_seg = sim.ticks_seg;

        /* cada evento usa DELTA ticks */
        sim.t_total = DELTA * sim.t_total;
        nodo0.timeout = DELTA * nodo0.timeout;
        nodo1.timeout = DELTA * nodo1.timeout;
        nodo0.ack_timeout = DELTA * nodo0.ack_timeout;
        nodo1.ack_timeout = DELTA * nodo1.ack_timeout;
        canal0.t_prop = DELTA * canal0.t_prop;
        canal1.t_prop = DELTA * canal1.t_prop;

        /* probabilidad de error */
        nodo0.prob_error = 10 * nodo0.prob_error;
        nodo1.prob_error = 10 * nodo1.prob_error;

        /* queremos acks  y nacks? */
        nodo0.nacks = sim.nacks;
        nodo1.nacks = sim.nacks;
        nodo0.acks = sim.acks;
        nodo1.acks = sim.acks;

        /* copia de la semilla */
        nodo0.semilla = sim.semilla;
        nodo1.semilla = sim.semilla;

        /* estos parámetros se utilizan para los cálculos */
        nodo0.tickseg = sim.ticks_seg;
        nodo1.tickseg = sim.ticks_seg;
        nodo0.rb = canal0.rb;
        nodo1.rb = canal1.rb;

        /* rellena la estructura args */
        args.simulacion = sim;
        args.canal0 = canal0;
        args.canal1 = canal1;
        args.worker0 = nodo0;
        args.worker1 = nodo1;

        return(args);    /* devuelve los parámetros ya que ho habido errores */
}

FILE *avanza(FILE *fich, char *buf)
/* lee la siguiente palabra saltandose los comentarios */
{
        fscanf(fich, "%s", buf);                        /* lee la siguiente palabra */

        while(buf[0] == '#') {                        /* salta líenas de comentario */
                fscanf(fich, "%[^\n]", buf);        /* las líeas de comentario empiezan por # */
```

```
		fscanf(fich, "%s", buf);
	}
	return(fich);
}


int iguales(char *uno, char *dos)
/* Compara dos cadenas de texto y devuelve un 1 si son iguales, sino un 0
 */
{
	int i=0;
	do {
		if(uno[i]!=dos[i]) return(0);
	} while(dos[i++]!='\0');
	return(1);
}

param_sim parse_sim(FILE *fich)
/* Lee los parámetros generales de la simulación una vez encontrada la palabra
 * "SIMULACION" en el fichero de entrada
 */
{
	param_sim sim;		/* estructura donde almacenar los parametros */
	char buf[100];	/* almacena lo que se lee */
	bigint baux;	/* variable auxiliar */

	fich = avanza(fich, buf);		/* lee la siguiente palabra */

	/* según la palabra que lee, la mete en la estructura de los parámetros y compreuba
	   que está entre los límites */
	do {
		if(iguales(buf, "tiempo_total")) {
			/* Tiempo total de simulacion en seguendos */
			fscanf(fich, "%d", &baux);
			sim.t_total = baux;
			if(sim.t_total < 0) {
				printf("El tiempo total de simulación ha de ser positivo.\n");
				exit(1);
			}
		} else if(iguales(buf, "ticks_seg")) {
			/* numero de ticks por segundo */
			fscanf(fich, "%d", &baux);
			sim.ticks_seg = baux;
			if (baux < 0 || baux == 0) {
				printf("El numero de ticks/seg no puede ser negativo ni nulo\n");
				exit(1);
			}
		} else if(iguales(buf, "semilla")) {
			/* Semilla utilizada para producir la secuencia pseudoaleatoria */
			fscanf(fich, "%d", &baux);
			sim.semilla = baux;
			if (baux < 0) {
				printf("La semilla ha de ser un número positivo\n");
				exit(1);
			}
		} else if(iguales(buf, "acks")) {
			/* asentimientos positivos sí (1) o no (!=1) */
			fscanf(fich, "%d", &baux);
			if(baux) sim.acks = 1;
			else sim.acks = 0;
		} else if(iguales(buf, "nacks")) {
			/* asentimientos negativos sí (1) o no (!=1) */
			fscanf(fich, "%d", &baux);
			if(baux) sim.nacks = 1;
			else sim.nacks = 0;
		} else if(iguales(buf, "fichero_salida")) {
			/* fichero donde se muestran los resultados */
			baux = 0;
			fscanf(fich, "%s", buf);
			/* copia la cadena del fichero en la estructura, caracter a caracter */
			do sim.fichero[baux] = buf[baux];
			while(buf[baux++]!='\0');
		} else {
			/* si no se ha leído ninguna de estas palabras, es que hay algún error */
			printf("Error en fichero de entrada: %s\n", buf);
			exit(1);
		}
		fich = avanza(fich, buf);		/* lee la siguiente palabra */
	/* la cadena "/SIMULACION" indica fin de los parámetros generales */
	} while(!iguales(buf, "/SIMULACION"));

	return(sim);	/* devuelve la estructura con los parámetros */
}

param_worker parse_nodo(FILE *fich)
/* Lee los parámetros de nodo de la simulación una vez encontrada la palabra
 * "NODO" en el fichero de entrada
 */
{
	param_worker nodo;	/* estructura donde almacenar los parametros */
	char buf[100];		/* almacena lo que se lee */
	bigint baux;		/* variable auxiliar */

	nodo.enviar = NULL;		/* en principio las cadenas de los ficheros están vacías */
	nodo.recibir = NULL;

	fich = avanza(fich, buf);		/* lee la siguiente palabra */

	/* según la palabra que lee, la mete en la estructura de los parámetros y compreuba
	   que está entre los límites */
	do {
		if(iguales(buf, "temporizador")) {
			/* Tiempo de expiracion del temporizador principal en ms */
			fscanf(fich, "%d", &baux);
			nodo.timeout = baux;
			if (baux < 0 || baux == 0){
				printf("temporizador ha de ser positivo.\n");
				exit(1);
			}
		} else if(iguales(buf, "temporizador_ack")) {
			/* Tiempo de expiracion del temporizador de ack en ms */
			fscanf(fich, "%d", &baux);
			nodo.ack_timeout = baux;
			//if (baux < 0 || baux == 0){
```

```
        if (baux < 0){
                printf("temporizador_ack debe ser positivo\n");
                exit(1);
        }
} else if(iguales(buf, "ventana_trx")) {
        /* ventana de transmision */
        fscanf(fich, "%d", &baux);
        nodo.v_trx = baux;
        if (baux < 0 || baux == 0){
                printf("La ventana de transmisión debe ser positiva\n");
                exit(1);
        }
} else if(iguales(buf, "ventana_rx")) {
        /* ventana de recepion */
        fscanf(fich, "%d", &baux);
        nodo.v_rx = baux;
        if (baux < 0 || baux == 0){
                printf("La ventana de recepción debe ser positiva\n");
                exit(1);
        }
} else if(iguales(buf, "mtu")) {
        /* tamaño máximo de trama en octetos */
        fscanf(fich, "%d", &baux);
        nodo.mtu = baux;
        if (baux <= 8 || baux > 100){
                printf("El MTU debe ser mayor que 8 y menor que 100 octetos.\n");
                exit(1);
        }
} else if(iguales(buf, "fichero_enviar")) {
        /* fichero a enviar */
        fscanf(fich, "%s", &buf);
        nodo.enviar = fopen(buf, "rt");
        if(nodo.enviar == NULL) {
                printf("Error al abrir fichero %s.\n", buf);
                exit(1);
        }
} else if(iguales(buf, "fichero_recibir")) {
        /* fichero a recibir */
        fscanf(fich, "%s", &buf);
        nodo.recibir = fopen(buf, "wt");
        if(nodo.recibir == NULL) {
                printf("Error al abrir fichero %s.\n", buf);
                exit(1);
        }
} else if(iguales(buf, "traza")) {
        /* Opciones de traza.  Los bits están definidos en worker.c. */
        fscanf(fich, "%d", &baux);
        nodo.debug_flags = baux;
        if (baux < 0) {
                printf("traza debe ser positivo.\n");
                exit(1);
        }
} else {
        printf("Error en fichero de entrada: %s\n", buf);
        exit(1);
}
fich = avanza(fich, buf);       /* lee la siguiente palabra */
        /* la cadena "/NODO" indica fin de los parámetros del nodo */
        } while(!iguales(buf, "/NODO"));

        return(nodo);   /* devuelve la estructura con los parámetros */
}

param_canal parse_canal(FILE *fich)
/* Lee los parámetros de canal de la simulación una vez encontrada la palabra
 * "CANAL" en el fichero de entrada
 */
{
        param_canal canal;      /* estructura donde almacenar los parametros */
        char buf[100];          /* almacena lo que se lee */
        bigint baux;            /* variable auxiliar */

        fich = avanza(fich, buf);       /* lee la siguiente palabra */

        /* según la palabra que lee, la mete en la estructura de los parámetros y compreuba
          que está entre los límites */
        do {
                if(iguales(buf, "regimen_binario")) {
                        /* regimen binario en bits por segundo */
                        fscanf(fich, "%d", &baux);
                        canal.rb = baux;
                        if (baux < 0 || baux == 0) {
                                printf("El regimen binario no puede ser negativo ni nulo\n");
                                exit(1);
                        }
                } else if(iguales(buf, "velocidad_propagacion")) {
                        /* velocidad de propagacion en metros por segundo*/
                        fscanf(fich, "%d", &baux);
                        canal.vp = baux;
                        if (baux < 0 || baux == 0) {
                                printf("La velocidad del propagacion en el canal\n
                                        no puede ser negativa ni nula\n");
                                exit(1);
                        }
                } else if(iguales(buf, "longitud")) {
                        /* longitud de la linea en metros */
                        fscanf(fich, "%d", &baux);
                        canal.l = baux;
                        if (baux < 0 || baux == 0) {
                                printf("La longitud del canal no puede ser negativa ni nula\n");
                                exit(1);
                        }
                } else if(iguales(buf, "error_trama")) {
                        /* Probabilidad de que una trama recibida tenga error */
                        fscanf(fich, "%d", &baux);
                        canal.prob_error = baux;
                        if (baux < 0 || baux > 100) {
                                printf("error_trama debe estar entre 0 y 100.\n");
                                exit(1);
                        }
                } else {
                        printf("Error en fichero de entrada: %s\n", buf);
                        exit(1);
                }
```

```
                fich = avanza(fich, buf);       /* lee la siguiente palabra */
        /* la cadena "/CANAL" indica fin de los parámetros del canal */
        } while(!iguales(buf, "/CANAL"));

        return(canal);  /* devuelve la estructura con los parámetros */
}

int calcula_max_seq(int v1, int v2, int v3, int v4)
/* Calcula el numero máximo de secuencia teniendo en cuenta los 4 numeros que
 * se le pasan como parámetros de entrada, que son los tamaños de las 4 ventanas
 */
{
        int aux;        /* variable auxiliar */
        int i = 0;      /* variable índice */

        /* aux = ventana de mayor tamaño */
        aux = v1;
        aux = (v2 > aux? v2 : aux);
        aux = (v3 > aux? v3 : aux);
        aux = (v4 > aux? v4 : aux);

        /* vantana mayor tamaño <= (max_seq+1)/2 */
        /* con max_seq = (2^n)-1 */
        do ;
        while(aux > exp2(i++));
        /* además max_seq + 1 ha de ser múltiplo de las ventanas de recepción */
        return(exp2(i)-1);
}

int exp2(int i)
/* calcula la i-ésima potencia de 2 */
{
        int j;
        int aux = 1;

        for(j = 0; j < i; j++) aux = aux * 2;
        return(aux);
}
```

7 Protocol.c

```
/* Este fichero contiene las funciones que desarrollan el protocolo de nivel de enlace
 * que se va a simular. Se ha procurado que el protocolo sea lo más versatil prosible, por
 * lo cual aparece un elevado número de parámetros y variables en la distintas funciones
 */

/* El origen de la mayoría de las funciones de este fichero son las funciones del protocolo 6
 * del simulador original, pero puede comprobarse que está bastante cambiadas.
 */

/* La ejecución comienza en la función protocol(), que es llamada por el main().
 * Esta función utiliza muchas funciones contenidas en worker.c.
 * Esta función ejecuta un bucle sin fin. Cada vuelta del bucle es un evento.
 * Este hilo finaliza cuando se llama a la función exit() desde la función wait_for_event().
 */

#include "common.h"
#include "protocol.h"

/* prototipos de las funciones */
static inc(seq_nr *num, int max);
static boolean between(seq_nr a, seq_nr b, seq_nr c);
static frame new_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected,
                                        packet buffer[], bigint t_trans[], param_worker args);
static nodo *send_frame(frame s, nodo *list, int id, timers *t,
                                        param_worker args, statistics *st);

void protocol(filedesc fd, param_worker args)
{
        /* Declaración de variables */
        int network_layer_status;       /* estado del nivel de red; 0 = deshabilitado, 1 = habilitado */
        unsigned int next_net_pkt;      /* no. de secuencia del siguiente paquete de red */
        unsigned int last_pkt_given;/* ultima paquete de red entregado en orden */
        seq_nr ack_expected;            /* extremo inferior de la ventana de transmisión */
        seq_nr next_frame_to_send;      /* extremo superior de la ventana de transmisión + 1 */
        seq_nr frame_expected;                  /* extremo inferior de la ventana de recepción */
        seq_nr too_far;                         /* extremo superior de la ventana de recepción  +1 */
        seq_nr aux_nr;                          /* variable auxiliar */
        int i;                                  /* variable índice */
        frame r;                                /* variable auxiliar */
        timers tm;                              /* estructura de los temporizadores */
        bigint tmrs[args.max_seq+1];/* tabla de temporizadores */
        seq_nr sqs[args.max_seq+1];  /* tabla de no. de secuencia de los temporizadores */
        packet out_buf[args.max_seq + 1];       /* buffers para las tramas transmitidas no asentidas */
        bigint t_trans[args.max_seq + 1];       /* tick en el que las tramas se transmitieron por
                                                                                primera vez */
        packet in_buf[args.max_seq + 1];        /* buffers para las tramas que llegan */
        boolean arrived[args.max_seq + 1];      /* indica que tramas del anterior buffer
                                                                                están ocupadas */
        boolean no_nak = true;                  /* aún no se ha transmitido ningún nack */
        seq_nr nbuffered;                       /* posiciones del buffer de salida ocupadas */
        event_type event;                       /* almacena el último evento */
        statistics stat;                /* almacena las estadísticas */
        buffer b;                               /* buffer donde se almacenan las tramas pasadas al nivel
                                                                fisico antes de ser transmitidas */
        frame inqueue[MAX_QUEUE*args.v_rx];     /* buffer intermedio de recepción */
        frame nfr;                              /* variable auxiliar */
        packet auxp;                            /* variable auxiliar */
        int nt_bufs = args.v_trx;       /* número de posiciones del buffer de transmisión */
        int nr_bufs = args.v_rx;        /* número de posiciones del buffer de recepción */
        int max_seq = args.max_seq;  /* número máximo de secuencia */
        boolean fin_fichero;            /* indica si se ha llegado al final del fichero a enviar */
        boolean fin_fich_rec;           /* indica si se ha recibido el final del fichero a recibir */
        boolean fin;                            /* indica si se ha llegado al final de la simulación */

        /* inicialización de variables */
        stat = inicia_estadisticas();           /* inicializa las estadísticas */
        stat.tickseg = args.tickseg;            /* estos parámetros se introducen en las estadísticas */
        stat.rb = args.rb;                              /* para facilitar los cálculos */
        tm.oldest_frame = max_seq + 1;                  /* valor inicial para el simulador */
        tm.ack_timer = tmrs;                            /* introduce la tabla de temporizadores principales
                                                                        en su estructura correspondiente */
        tm.seqs = sqs;                                  /* introduce la tabla de no. de secuencia de
```

```
                                                        temporizadores en su estructura
correspondiente */
        tm.aux_timer = 0;                               /* inicializa el temporizador de asentimiento a 0 */
        tm.tick = 0;                                    /* el tick inicial el 0 */                          args.mtu);
        tm.event = NO_EVENT;                            /* el primer evento es no_event */
        tm.lowest_timer = 0;                    /* el temporizador más bajo está a 0 */
        ack_expected = 0;                               /* numero de secuencia del asentimiento esperado */
        next_frame_to_send = 0;                         /* numero de secuencia de la siguente trama a enviar */
        frame_expected = 0;                             /* trama esperada */
        too_far = args.v_rx;                    /* extremo superior de la ventana de recepción  +1 */
        nbuffered = 0;                                  /* inicialmente no hay tramas en el buffer */
        b.queue = inqueue;
        b.in = 0;
        b.out = 0;
        b.nframes = 0;
        b.list = NULL;
        b.max_seq = max_seq;
        b.trans = true;
        tm.oldest_frame = 0;
        fin = true;                                             /* todavía no se ha llegado al final de la simulación
*/
        tm.nr_timers = max_seq + 1;             /* establece el número de temporizadores principales */
        /* inicializa las tablas de los temprizadores */
        for(i = 0; i <= max_seq + 1; i++) arrived[i] = false;
        for(i = 0; i < tm.nr_timers; i++) {
                tm.seqs[i] = 0;
                tm.ack_timer[i] = 0;
        }

        /* inicializa las variables de la capa de red  */
        network_layer_status = enable_network_layer();
        next_net_pkt = 0;
        last_pkt_given= 0xFFFFFFFF;

        /* utiliza la semilla y le suma el indice del nodo para que cada nodo tenga una diferente */
        srand(args.semilla + fd.id);

        /* comprueba protocolo unilateral */
        if (args.recibir == NULL) fin_fich_rec = true;
        else fin_fich_rec = false;
        if (args.enviar == NULL) {
                network_layer_status = disable_network_layer();
                fin_fichero = true;
        } else fin_fichero = false;



        while (true) {

                b.frm_exp = frame_expected;
                /* espera a que se produzca un evento, hay cinco posibilidades */
                event = wait_for_event(fd, &tm, network_layer_status, &b, &fin_fich_rec,
                        args, &stat);
                switch(event) {                 /* actúa según el evento */
                        case network_layer_ready:       /* el nivel de red tiene un paquete nuevo */
                                nbuffered = nbuffered + 1;      /* expande la ventana de transmisión */
                                /* coge el nuevo paquete y comienza a procesarlo */
                                /* lo almacena en el buffer de transmisión, en la posición next_frame_to_send */
                                out_buf[next_frame_to_send] = from_network_layer(&next_net_pkt,
                                                                        args.enviar, &fin_fichero,
                                /* guarda en que tick se comenzó a procesar el nuevo paquete */
                                t_trans[next_frame_to_send] = tm.tick;
                                /* crea una nueva trama con el nuevo paquete */
                                nfr = new_frame(data, next_frame_to_send, frame_expected, out_buf,
                                        t_trans, args);
                                /* manda la trama al nivel fisico */
                                b.list = send_frame(nfr, b.list, fd.id, &tm, args, &stat);
                                /* avanaza el extremo superior de la ventana de transmisión */
                                inc(&next_frame_to_send, max_seq);
                                break;

                case frame_arrival:             /* se recibe una trama del nivel fisico */
                        from_physical_layer(&r, b);     /* coge la nueva trama y la almacena en r */
                        if (r.kind == data) {           /* la trama es de datos y correcta */
                                /* si su numero de secuencia es distinto de frame_expected quiere decir
                                   que llega fuera de secuencia */
                                if ((r.seq != frame_expected) && no_nak && args.nacks) {
                                        /* si además aún no se ha mandado ningun nack y estos estan
                                           permitidos se genera un nack */
                                        no_nak = false;                 /* solo un nacl por trama */
                                        /* se genera la nieva trama de tipo nack */
                                        nfr = new_frame(nak, 0, frame_expected, 0, 0, args);
                                        /* se manda la nueva trama al nivel fisico */
                                        b.list = send_frame(nfr, b.list, fd.id, &tm, args, &stat);
                                /* si la trama llega en secuencia comienza el temp. de asentimiento */
                                } else if(tm.aux_timer == 0) start_ack_timer(&tm, args.ack_timeout);
                                /* si la trama cae dentro de la ventana de recepción, se acepta, sino
                                   se ignora */
                                if (between(frame_expected, r.seq, too_far)
                                        && (arrived[r.seq] == false)) {
                                        /* actualiza las estadísticas, primero el retardo */
                                        stat.delay += (tm.tick - r.t_trans) /DELTA;
                                        /* y de cuantas tramas se ha contabilizado el retardo */
                                        stat.delay_counted++;
                                        /* marca la posición del buffer de recepción como ocupada */
                                        arrived[r.seq] = true;
                                        in_buf[r.seq] = r.info;  /* inserta el paquete en el buffer */
                                        /* actualiza el numero de bytes recibidos */
                                        stat.bytes_trx += strlen(r.info.linea) * sizeof(char);
                                        /* pasa los paquetes al nivel de red en orden, si estan dispuestos */
                                        while (arrived[frame_expected]) {
                                                /* saca el paquete del buffer de recepción */
                                                auxp = in_buf[frame_expected];
                                                arrived[frame_expected] = false;
                                                /* pasa el paquete al nivel de red */
                                                last_pkt_given =
                                                        to_network_layer(auxp, last_pkt_given, fd.id,
                                                        tm.tick, args, &stat);
                                                no_nak = true;          /* permite que vuelvan a enviarse nacks */
                                                /* avanza el extremo inferior de la ventana de recepción */
                                                inc(&frame_expected, max_seq);
                                                /* avanza el extremo superior de la ventana de recepción */
                                                inc(&too_far, max_seq);
```

```
                                        /* si aún no ha comenzado el temp. de asentimeinto, lo inicia */
                                        if(tm.aux_timer == 0) start_ack_timer(&tm, args.ack_timeout);
                                        /* era el último paquete del fichero? */
                                        if(last_pkt_given == 0) fin = !fin;
                                        if(fin) fin_fich_rec = true;
                                }
                        }
                }
                /* la trama es un nack y está dentro de la ventana de transmisión */
                if((r.kind==nak) &&
                        between(ack_expected,(r.ack+1)%(max_seq+1),next_frame_to_send)) {
                        /* genera de nuevo la trama que pide el nack */
                        nfr = new_frame(data, (r.ack+1) % (max_seq + 1), frame_expected,
                                out_buf, t_trans, args);
                        /* manda la trama al nivel fisico */
                        b.list = send_frame(nfr, b.list, fd.id, &tm, args, &stat);
                }
                /* procesa el campo de asentimiento de la trama recibida */
while (between(ack_expected, r.ack, next_frame_to_send)) {
                        /* el asentimiento está dentro de los esperados */
                        nbuffered = nbuffered - 1;      /* saca una trama del buffer de transmision */
                        stop_timer(ack_expected, &tm);          /* para el temp. de esa trama */
                        /* actualiza las estadísticas de retardo medio de vuelta */
                        stat.round_delay += (tm.tick - t_trans[ack_expected])/DELTA;
                        stat.trip_cuonted++;
                        /* avanza el extremo inferior de la ventana de transmisión */
                        inc(&ack_expected, max_seq);
                }
                break;

        case cksum_err:                 /* llega una trama con error */
                /* si se pueden mandar nacks mando uno */
                if (args.nacks) {
                        no_nak= false;                  /* solo un nack por trama */
                        /* genera el nack */
                        nfr = new_frame(nak, 0, frame_expected, out_buf, 0, args);
                        /* manda el nack al nivel fisico */
                        b.list = send_frame(nfr, b.list, fd.id, &tm, args, &stat);
                }
                break;

        case timeout:           /* expira un temporizador */
                /* genera de nuevo la trama a retransmitir */
                nfr = new_frame(data, tm.oldest_frame, frame_expected, out_buf,
                                t_trans, args);
                /* manda la trama al nivel fisico */
                b.list = send_frame(nfr, b.list, fd.id, &tm, args, &stat);
                break;

        case ack_timeout:       /* expira el temporizador de asentimiento */
                if(args.acks) {
                        /* genera una trama de asentimiento */
                        nfr = new_frame(ack,0,frame_expected, 0, 0, args);
                        /* manda la nueva trama al nivel fisico */
                        b.list = send_frame(nfr, b.list, fd.id, &tm, args, &stat);
                }
                break;
                }
                /* si el numero de tramas en el buffer de transmisión es menor que el tamaño del
                   buffer y aún no se ha llegado al final del fichero a enviar, se habilita
                   el nivel de red */
                if (nbuffered < nt_bufs && !fin_fichero)
                        network_layer_status = enable_network_layer();
                /* si alguna de las dos cosas no se da, se dashabilita el nivel de red, que no
                   puede generar más tramas */
                else network_layer_status = disable_network_layer();
        }
}

static inc(seq_nr *num, int max)
{
/* Incrementa num circularmente, con max como máximo.
 */
        if (*num < max) *num = *num + 1;
        else *num = 0;
}

static boolean between(seq_nr a, seq_nr b, seq_nr c)
{
/* Devuelve true si b se encuentra entre a y c, pero circularmente.
 */
  return ((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));
}

static frame new_frame(frame_kind fk, seq_nr frame_nr, seq_nr frame_expected,
                                                packet buffer[], bigint t_trans[], param_worker args)
{
/* Contruye una nueva trama a partir de los datos de sus parámetros
 */
        frame s;        /* trama a formar */

        s.kind = fk;    /* tipo de dato; kind == data, ack, o nak */
        /* si es de tipo datos, se rellena su campo info, y el tiempo en que se transmitió
           por primera vez (t_trans) */
        if (fk == data) {
                s.info = buffer[frame_nr];
                s.t_trans = t_trans[frame_nr];
        }

        /* si la trama no es de datos rellena los campos de información com basura pero de
           manera que luego quede bien en la traza */
        if (fk == nak || fk == ack) {
                s.info.data[0] = 1;
                s.info.data[1] = 1;
                s.info.data[2] = 1;
                s.info.data[3] = 1;
                s.info.linea[0] = '  \0'  ;
        }
        s.seq = frame_nr;       /* solo tiene sentido en tramas de datos, numero de secuencia */
        /* el campo ack lleva el numero de secuencia de la ultima trama recibida sin error */
        s.ack = (frame_expected + args.max_seq) % (args.max_seq + 1);
        return(s);      /* devuelve la trama creada */
}
```

```
static nodo *send_frame(frame s, nodo *list, int id, timers *t,
                                        param_worker args, statistics *st)
{
/* Manda una trama al nivel fisico */
        list = to_physical_layer(s, list, id, t->tick, args, st);     /* al nivel fisico */
        if (s.kind == data) {     /* si la trama es de datos inicia el temporizador correspondiente */
                start_timer(s.seq, args.timeout, t);
                t->seqs[s.seq] = s.seq;                 /* tb hay que guardar el numero de secuencia de la trama */
        }
        stop_ack_timer(&(t->aux_timer));      /* reinicializa el temporizador de asentimiento */
        /* devuelve el puntero a la lista de tramas enviadas al nivel físico pero que aún no han
           sido transmitidas */
        return(list);
}
```

8 Worker.c

```
/* Este fichero contiene el codigo correspondiente a las funciones utilizadas por los nodos
 * para implementar el protocolo de nivel de enlace.
 * Todas las funciones de este fichero son llamadas por funciones del fichero p.c o por
 * funciones de este mismo fichero
 */

#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <unistd.h>
#include "common.h"
#include "hilossec.h"

#define BYTE 0377                       /* mascara de byte */
#define UINT_MAX  0xFFFFFFFF    /* valor máximo de unsigned int */
#define INTERVAL 100000                 /* intervalo para la traza periódica */

/* prototipos */
event_type wait_for_event(filedesc fd, timers *t, int n_l_s, buffer *b,
                                        boolean *fin_fich_rec, param_worker args, statistics *st);
void queue_frames(filedesc fd, buffer *b, int v_rx, statistics st);
int pick_event(filedesc fd, int network_layer_status, timers *t, buffer *b,
                        param_worker args, statistics *st);
event_type frametype(int id, bigint tick, buffer *b, param_worker args, statistics *st);
packet from_network_layer(unsigned int *next_net_pkt, FILE *fich, boolean *eof, int mtu);
unsigned int to_network_layer(packet p, unsigned int last_pkt_given, int id,
                                        bigint tick, param_worker args, statistics *st);
void from_physical_layer(frame *r, buffer b);
nodo *to_physical_layer(frame s, nodo *list, int id, bigint tick,
                                        param_worker args, statistics *st);
void start_timer(seq_nr k, bigint timeout_interval, timers *t);
void stop_timer(seq_nr k, timers *t);
void start_ack_timer(timers *t, bigint ack_timeout);
void stop_ack_timer(bigint *aux_timer);
int enable_network_layer(void);
int disable_network_layer(void);
int check_timers(timers *t);
int check_ack_timer(bigint *timer, bigint time);
unsigned int pktnum(packet p);
char *fr(frame f, char *frm);
bigint recalc_timers(bigint timer[], int nr);
void print_statistics(filedesc fd, statistics st);
void sim_error(char *s, int fd);
void manda_bandera(filedesc fd, statistics st);
statistics inicia_estadisticas(void);
nodo *enlista(frame fr, nodo *list);
void comprueba_lista(buffer *b, timers *t, filedesc fd, statistics st, bigint tout);
void itoa(int num, char *cad, int base);
void print_traza(traza_kind tipo, bigint tick, int id, frame f, seq_nr seq,
                                        statistics st);

event_type wait_for_event(filedesc fd, timers *t, int n_l_s, buffer *b,
                                                boolean *fin_fich_rec, param_worker args, statistics *st)
/* Esta función lee el tick del main y compueba el estado del nodo (transmisor, capa de red,
 * temporizadores..) para ver cual es el siguiente evento a procesar. Además pasa el tick
 * a su canal y actualiza estadísticas
 */
{
        frame frm;                      /* trama en proceso */
        bigint ct;                      /* almacena el tiempo leido del main */
        bigint word = OK;       /* respuesta enviada al main */
        do {  /* espera un evento y encola las tramas que llegan */
                if(t->event == NO_EVENT) {          /* se han acabado los eventos del anterior tick */
                        /* previene varios eventos en el mismo tick, aunque ahora no es problema */
                        t->offset = 0;
                        /* comprueba si hay tamas nuevas que recibir */
                        queue_frames(fd, b, args.v_rx, *st);
                        /* cheque el estado del transmisor */
                        comprueba_lista(b, t, fd, *st, args.timeout);
                        /* si recibe fin fichero lo comunica a Main */
                        if(*fin_fich_rec) {
                                word = END_FILE;
                                *fin_fich_rec = false;
                        }
                        /* manda el estado al Main */
                        if (write(fd.mw, &word, TICK_SIZE) != TICK_SIZE) print_statistics(fd, *st);
                        /* lee el tiempo del main */
                        if (read(fd.mr, &ct, TICK_SIZE) != TICK_SIZE) print_statistics(fd, *st);
                        /* escribe el tick al canal */
                        st->ttotal = t->tick / DELTA;
                        if (write(fd.cw, &ct, TICK_SIZE) != TICK_SIZE) print_statistics(fd, *st);
                        /* si el tick recibido es 0, imprime las estadisticas y finaliza */
                        if (ct == 0) print_statistics(fd, *st);
                        t->tick = ct;               /* actualiza el tiempo */
                }

                /* ahora elige un evento */
                t->event = pick_event(fd, n_l_s, t, b, args, st);
                if (t->event == NO_EVENT) {                     /* no hay ningún evento */
                        /* si además todos los temporizadores están parados, el nodo está inactivo y
                           se manda NOTHING, si hay algun temporizador funcionando se manda OK, esto
                           se hace para evitar bucles infinitos */
                        word = (t->lowest_timer == 0 ? NOTHING : OK);
                        /* se se han establecido trazas periódicas y se está en el tick preciso, se
                           imprime la traza periódica */
```

```
                if ((args.debug_flags & PERIODIC) && (t->tick%INTERVAL == 0))
                        print_traza(periodic, t->tick, fd.id, frm, 0, *st);
        }
    /* si no hay ningún evento se repite el bucle, es decir, no se vuelve a la función
       protocol() en cada tick, sino cuando se produce un evento */
    } while (t->event == NO_EVENT);

    // word = OK;
    /* actualiza las estadísticas correspondientes a los temporizadores principales */
    if (t->event == timeout) {
        st->timeouts++;
        st->data_retransmitted++;
        /* imprime la traza correspondiente */
        if (args.debug_flags & TIMEOUTS)
            print_traza(tmout, t->tick, fd.id, frm, t->oldest_frame, *st);
    }

    /* actualiza las estadísticas correspondientes alo temporizador de asentimiento */
    if (t->event == ack_timeout) {
        st->ack_timeouts++;
        /* imprime la traza correspondiente */
        if (args.debug_flags & TIMEOUTS)
            print_traza(ack_tmout, t->tick, fd.id, frm, 0, *st);
    }
    /* devuelve el evento que se ha producido para su procesamiento */
    return(t->event);
}


void queue_frames(filedesc fd, buffer *b, int v_rx, statistics st)
/* Comprueba si ha llegado alguna trama, si es así las mete en cola que es un buffer circular.
 * En el programa original se utilizaba la función fstat() que devolvía el estado de un pipe,
 * en particular su tamaño, para comprobar si se había escrito algo. Pero esta función aunque
 * está incluída en las librerías de Cygwin, no funciona, por lo que se ha recurrido a un
 * truquillo para comprobar si se ha escrito algo en el pipe.
 * Lo que se hace es, antes de leer el pipe se introduce en él una trama de tipo end,
 * que es la bandera; después se lee el pipe hasta leer la bandera, que es el final del pipe.
 * Esto se hace así porque si se lee de un pipe vacío el programa se queda colgado esperando.
 */
{
    frame aux;      /* para almacenar la trama que se lee */

    manda_bandera(fd, st);          /* escribe la bandera */

    while(true) {
        /* lee una trama del pipe y la guarda en aux */
        if (read(fd.wr, &aux, FRAME_SIZE) != FRAME_SIZE)
            sim_error("Error reading frames from peer", fd.mw);
        if(aux.kind == end) break;  /* es la bandera, el pipe está vacío */
        /* almacena la trama en la primera posicion libre del buffer */
        b->queue[b->in] = aux;
        b->in++;        /* avanza el índice de la primera posición libre */
        b->nframes++;           /* numero de tramas en el buffer */
        /* tamaño del buffer: n * ventana de recepción */
        /* pero este valor se puede cambiar si hace falta */
        if (b->nframes >= MAX_QUEUE * v_rx)     /* comprueba si está lleno el buffer */
            sim_error("Out of queue space. Increase MAX_QUEUE and re-make.", fd.mw);
        if (b->in == MAX_QUEUE * v_rx) b->in = 0;       /* esto hace el buffer circular */
    }
}


int pick_event(filedesc fd, int network_layer_status, timers *t, buffer *b,
                param_worker args, statistics *st)
/* Elige el siguiente evento que se debe procesar a continuación, si hay alguno posible.
 * El orden en que se comprueban los posibles eventos es importante ya que da prioridad a
 * unos eventos sobre otros.
 */
{
    /* eventos posibles: {frame_arrival, cksum_err, timeout, net_rdy, ack_timeout} */
    /* si el tramsmisor está ocupado solo son posibles frame_arrival y cksum_error */
    /* primero comprueba el temporizador de asentimiento */
    if (check_ack_timer(&(t->aux_timer), t->tick) > 0) return(ack_timeout);
    /* si hay tramas en el buffer de recepción se produce un frametype */
    if (b->nframes > 0) return((int) frametype(fd.id, t->tick, b, args, st));
    /* transmisor ocupado: b->trans = false */
    /* hay tramas esperando a ser transmitidas: b->list != NULL */
    /* evento network_layer_status: el paquete de red tiene un paquete listo */
    if (network_layer_status && b->trans && (b->list == NULL))
        return(network_layer_ready);
    /* expira algún temporizador de retransmision */
    if (check_timers(t) >= 0) return(timeout);
    /* si se llega hasta aquí es que no se ha producido ningún evento */
    return(NO_EVENT);
}


event_type frametype(int id, bigint tick, buffer *b, param_worker args, statistics *st)
/* Esta función es llamada después de decidir que el siguiente evento es que llega una trama.
 * Pero queda por ver si esa trama será correcta, produciendo un evento frame_arrival, o
 * será errónea, produciendo en evento cksum_error.
 * Si no hay ningún error se saca al trama del buffer de recepción y se copia en last_frame.
 * Además se actualizan las estádisticas según los posibles casos.
 */
{
    int n;                                      /* almacena un número aleatorio */
    int i;                                      /* variable índice */
    event_type event;                           /* evento que se devuelve */
    int delay;                                  /* solo para mostrar por la salida */
    frame last_frame;                           /* trama que se saca del buffer */

    event = NO_EVENT;           /* inicializa la variable a NO_EVENT, por si acaso */
    /* saca una trama del buffer */
    last_frame = b->queue[b->out];      /* saca la primera trama de la pila LIFO */
    b->out++;               /* avanza el índice que apunta a la primera trama de la pila */
    if (b->out == MAX_QUEUE*args.v_rx) b->out = 0;  /* hace la pila circular */
    b->nframes--;           /* actualiza el número de tramas en el buffer */

    /* genera tramas con errores de checksum aleatorios */
    /* rand/RAND_MAX es menor que 1 y como prob_error está entre 0 y 1000, hay que
   multiplicarlo por 1000 */
    n = 1000 * (float)((float)rand()/(float)RAND_MAX);
    /* se ha detectado que generalmente la tasa de error suele ser levemente superior a la
       que se espera, no pudiendose corregir este comportamiento */
```

```
        if (n < args.prob_error) {      /* trama con error de checksum*/
                if (last_frame.kind == data) {  /* además es una trama de datos */
                        st->cksum_data_recd++;          /* actualiza estadísticas */
                        event = cksum_err;              /* genera un chsum_err */
                }
                if (last_frame.kind == ack) {   /* es una trama de asentimiento */
                        st->cksum_acks_recd++;          /* actualiza estadísticas pero no es necesario procesar
                                                                el error como si fuera de datos, genera
NO_EVENT */
                }
                i = 0;   /* indica trama con error */
        } else {        /* la trama es correcta */
                event = frame_arrival;          /* genera un frame_arrival */
                if (last_frame.kind == data) st->good_data_recd++;   /* actualiza estadísticas*/
                if (last_frame.kind == ack) st->good_acks_recd++;
                i = 1;   /* indica trama sin error */
        }
        if (args.debug_flags & RECEIVES) {          /* genera la traza si es necesario */
                /* calcula el retardo en el trayecto de la trama, pero solo si es de datos */
                if(last_frame.kind == data) delay = (tick - last_frame.t_trans)/DELTA;
                else delay = 0;
                /* imprime la traza */
                print_traza(( i == 0 ? bad_rec : good_rec), tick, id, last_frame, delay, *st);
        }
        b->last_frame = last_frame;    /* copia la variable local a la estructura */
        return(event);  /* devuelve el evento generado */
}


packet from_network_layer(unsigned int *next_net_pkt, FILE *fich, boolean *eof, int mtu)
/* recoge un paquete del nivel de red para transmitirlo por el canal.
 * Lee una línea del fichero a enviar, si encuentra fin de fichero (EOF)
 * devuelve un 0 en next_net_pkt
 */
{
        packet p;                       /* almacena el paquete que se genera */
        char *aux;                      /* apunta a la cadena leída */
        unsigned int num;       /* almacena el número de paquete */

        aux = fgets(p.linea, mtu, fich);        /* lee una línea del fichero de entrada */
        if(aux == NULL) {                               /* ha habido algún problema */
                if(fgetc(fich) == EOF) {                /* encontrado EOF */
                        num = 0;
                        *eof = true;                            /* para que se entere la función que le llama */
                } else {                                        /* error al leer el fichero */
                        printf("Error al leer fichero a enviar.\n");
                        exit(1);
                }
        } else num = *next_net_pkt;             /* no hay problema */
        /* además de la línea del fichero, en el paquete se incluye el número de paqute */
        p.data[0] = (num >> 24) & BYTE;
        p.data[1] = (num >> 16) & BYTE;
        p.data[2] = (num >>  8) & BYTE;
        p.data[3] = (num      ) & BYTE;

        *next_net_pkt = num + 1;        /* aumena el número de paquete para el siguiente paquete */
        return(p);                                      /* devuelve el paquete */
}


unsigned int to_network_layer(packet p, unsigned int last_pkt_given, int id,
                                                        bigint tick, param_worker args, statistics *st)
/* Entrega la información contenida en una trama recibida al nivel de red.
 * Hace un acomprobación para ver si el paquete está en secuencia, sino la simulación se
 * aborta con un mensaje de     "error de protocolo".
 */
{
        unsigned int num;       /* numero de secuencia del paquete */
        frame frm;                      /* trama ficticia para pasar a print_traza() */

        frm.info = p;           /* mete el paquete en la trama ficticia */
        num = pktnum(p);        /* lee el número de secuencia del paquete */
        if (num != last_pkt_given + 1) {        /* comprueba si el paquete está en secuencia */
                if(num != 0) {          /* error al entregar paquete */
                        /* genera un mensaje de error para saber cual ha sido el problema */
                        printf("Tick %u. Nodo %d error de protocolo.  Paquete entregado fuera de orden.\n",
                                tick/DELTA, id);
                        printf("Paqeute esperado %d pero obtuvo paquete %d\n", last_pkt_given+1,num);
                        exit(0);
                } else if(args.debug_flags & DELIVERS)      /* recibido último paquete de fichero */
                        print_traza(del_pkt, tick, id, frm, 0, *st);        /* imprime la traza */
        } else if(args.debug_flags & DELIVERS) {            /* no hay error al entregar paquete */
                if(num != 0) print_traza(del_pkt, tick, id, frm, 0, *st);          /* imprime la traza */
                else print_traza(del_ini, tick, id, frm, 0, *st);
        }
        /* escribe la línea recibida en el fichero correspondiente */
        if(fputs(p.linea, args.recibir) < 0) {
                printf("Error al escribir en fichero recibido.\n");
                exit(1);
        }
        /* actualiza estadísticas */
        st->payloads_accepted++;
        return(num);    /* devuelve el número de secuencia */
}


void from_physical_layer (frame *r, buffer b)
/* Saca la nueva trama recibida del buffer y la devuelve como parámetro de salida
 * Esta función no es estrictamente necesaria pero se conserva por similitud con el
 * programa original.
 */
{
 *r = b.last_frame;
}


nodo *to_physical_layer(frame s, nodo *list, int id, bigint tick,
                                        param_worker args, statistics *st)
/* Pasa la trama al nivel físico para su transmisión por el canal al otro nodo.
 * En realidad introduce la frama en una lista de tramas pendientes de ser tramsitidas.
 * Las tramas de esa lista las manda al transmisor la función comprueba_lista conforme
 * el transmisor se queda libre.
 * También actualiza las estadísticas de tramas enviadas.
 * En el programa original en esta función también se generaba estadísticamente otro
```

```
 * error, pero ahora esa funte de error se ha desestimado por no considerarla de interés.
 */
{
        /* actualiza las estadísticas */
        if (s.kind == data) st->data_sent++;
        if (s.kind == ack) st->acks_sent++;

        list = enlista(s, list);     /* mete la trama en la lista a la espera de ser transmitida */

        /* imprime la traza correspondiente */
        if (args.debug_flags & SENDS) print_traza(send_pkt, tick, id, s, 0, *st);
        return(list);       /* devuelve el puntero a la lista de tramas pendientes del transmisor */
}


void start_timer(seq_nr k, bigint timeout_interval, timers *t)
/* Arranca el temporizador de una trama de datos.
 * Lo que hace es poner en la entrada del temporizador el tick en el que ha de expirar el
 * temporizador de esa trama. También es importante guardar el núemro de secuencia de la trama
 * que ha iniciado el temporizador por si hay que retransmitirla.
 */
{
        /* calcula el tick en el que ha de expirar el temporizador y lo guarda en su lugar
           correspondiente */
        t->ack_timer[k] = t->tick + timeout_interval + t->offset;
        /* incrementa el offset para que no puedan expirar varios temporizadores en el mismo
           tick. Esto ya no es necesario, pues ya se pueden producir varios eventos en el
           mismo tick, pero en el programa original sí era necesario y por eso se conserva */
        t->offset++;
        /* busca el tick en el que expira el temporizador más proximo a expirar */
        t->lowest_timer = recalc_timers(t->ack_timer, t->nr_timers);
}


void stop_timer(seq_nr k, timers *t)
/* Para el temporizador correspondiente a la posición k.
 * Lo que hace es poner su entrada correspondiente a 0.
 */
{
        /* para el temporizador en cuestión */
        t->ack_timer[k] = 0;
        /* busca el tick en el que expira el temporizador más proximo a expirar */
        t->lowest_timer = recalc_timers(t->ack_timer, t->nr_timers);
}


void start_ack_timer(timers *t, bigint ack_timeout)
/* Arranca el temporizador de asentimiento.
 * Lo que hace es poner en la entrada del temporizador el tick en el que ha de expirar el
 * temporizador.
 */
{
        /* calcula el tick en el que ha de expirar el temporizador y lo guarda en su lugar
           correspondiente */
        t->aux_timer = t->tick + ack_timeout;
        /* incrementa el offset para que no puedan expirar varios temporizadores en el mismo
           tick. Esto ya no es necesario, pues ya se pueden producir varios eventos en el
           mismo tick, pero en el programa original sí era necesario y por eso se conserva */
        t->offset++;
}


void stop_ack_timer(bigint *aux_timer)
/* Para el temporizador de asentimiento.
 * Lo que hace es poner su entrada a 0.
 */
{
        /* para el temporizador de asentimiento */
        *aux_timer = 0;
}


int enable_network_layer(void)
/* Habilita la capa de red.
 * Lo único que hace es devolver un 1 que se almacenará en la variable network_layer_satus.
 * Esto permite que se generen más eventos network_layer_ready.
 */
{
        return(1);
}


int disable_network_layer(void)
/* Deshabilita la capa de red.
 * Lo único que hace es devolver un 0 que se almacenará en la variable network_layer_satus.
 * Esto impedirá que se generen más eventos network_layer_ready.
 */
{
        return(0);
}


int check_timers(timers *t)
/* Compueba los temporizadores por si le toca expirar a alguno.
 * Debido al uso del offset solo es posible que expire un temporizador a la vez, pero si
 * no se utilizase el offset no habría tampoco ningún problema, pues pueden procesarse dos
 * eventos timeout en el mismo tick.
 */
{
        int i;     /* variable índice */

        /* Mira si es posible que expire algún tenporizador en este tick, si no es posible
           devuelve un -1 */
        /* si lowest_timer es 0 quere decir que todos los temporizadores están parados */
        /* si lowest_timer es menor que el tick actual quiere decir que se ha pasado el tick
           en el que algún temporizador debía expirar y no se ha procesado, esto es un error */
        /* si el tick actual es menor que lowest_timer, aún no se puede expirar ningún
           temporizador */
        if (t->lowest_timer == 0 || t->tick < t->lowest_timer) return(-1);

        /* busca el temporizador que ha expirado */
        for (i = 0; i < t->nr_timers; i++) {
                if (t->ack_timer[i] == t->lowest_timer) { /* este es el temporizador que ha expirado */
                        t->ack_timer[i] = 0;     /* para el temporizador */
```

```
                        /* busca el tick en que expira el siguiente temporizador y lo almacena en
                           lowest_timer */
                        t->lowest_timer = recalc_timers(t->ack_timer, t->nr_timers);
                        /* busca el número de secuencia de la trama que inició el temporizador */
                        t->oldest_frame = t->seqs[i];
                        return(i); /* devuelve el número del temporizador que ha expirado */
                }
        }
        /* la ejecución del programa no puede llegar hasta aquí */
        printf("Imposible.  check_timers falló en %d\n", t->lowest_timer);
        exit(1);
}


int check_ack_timer(bigint *timer, bigint time)
/* Mira si ha expirado el temporizador de asentimiento.
 */
{
        if (*timer > 0 && time >= *timer) {
                /* el temporizador está arrancado y es mayor o igual al tick actual */
                *timer = 0;             /* para el temporizador */
                return(1);              /* devuelve un 1 indicando que sí ha expirado */
        } else {
                return(0);              /* devuelve un 0 indicando que no ha expirado */
        }
}


unsigned int pktnum(packet p)
/* Extrae el número de secuencia de un paquete y lo devuelve.
 */
{
        unsigned int num, b0, b1, b2, b3;

        b0 = p.data[0] & BYTE;
        b1 = p.data[1] & BYTE;
        b2 = p.data[2] & BYTE;
        b3 = p.data[3] & BYTE;
        num = (b0 << 24) | (b1 << 16) | (b2 << 8) | b3;
        return(num);
}


char *fr(frame f, char *frm)
/* Crea una cedena de carateres con la información de la trama para la traza.
 */
{
        char num[10];  /* cadena intermedia donde almacenar los números pasados a cadenas */
        char *tag[] = {"Data", "Ack ", "Nak "};         /* tipo de trama */

        strcat(frm, "(");           /* la cadena comenza con '  ('    */

        /* si la trama es de datos... */
        if(f.kind == data) {
                strcat(frm, tag[f.kind]);           /* lo primero es añadir el tipo de trama */
                strcat(frm, ", ");                          /* después va una '  ,'    */
                num[0] = '  \0'  ;                          /* vacía la cadena num */
                itoa(f.seq, num, 10);           /* en num pone el numero de secuencia en caracteres */
                strcat(frm, num);                           /* lo añade a la cadena */
                strcat(frm, ", ");                          /* otra '  ,'    */
                num[0] = '  \0'  ;                          /* vacía la cadena num */
                itoa(f.ack, num, 10);           /* en num pone el asentimiento en caracteres */
                strcat(frm, num);                           /* lo añade a la cadena */
                strcat(frm, ", ");                          /* otra '  ,'    */
                num[0] = '  \0'  ;                          /* vacía la cadena num */
                itoa(pktnum(f.info), num, 10);          /* en num pone el el número del paquete */
                strcat(frm, num);                           /* lo añade a la cadena */
                strcat(frm, ")");                           /* termina la cadena con '  )'    */
        }
        /* la trama no es de datos */
        else {
                strcat(frm, tag[f.kind]);           /* lo primero es añadir el tipo de trama */
                strcat(frm, ", ");                          /* después va una '  ,'    */
                num[0] = '  \0'  ;                          /* vacía la cadena num */
                itoa(f.seq, num, 10);           /* en num pone el numero de secuencia en caracteres */
                strcat(frm, num);                           /* lo añade a la cadena */
                strcat(frm, ", ");                          /* otra '  ,'    */
                num[0] = '  \0'  ;                          /* vacía la cadena num */
                itoa(f.ack, num, 10);           /* en num pone el asentimiento en caracteres */
                strcat(frm, num);                           /* lo añade a la cadena */
                strcat(frm, ")");                           /* termina la cadena con '  )'    */
        }
        return(frm);    /* devuelve la cadena creada */
}

bigint recalc_timers(bigint timer[], int nr)
/* Busca el temporizador que antes ha de expirar y devuelve en que tick lo hará.
 */
{
        int i;                                  /* variable índice */
        bigint t = UINT_MAX;            /* pone en t el valor más grande posible para un bigint */

        /* recorre los temporizadores */
        for (i = 0; i < nr; i++) {
                /* si este temporizador está arrancado lo compara con el más bajo, y si es menor,
                   almacena su valor en t */
                if (timer[i] > 0 && timer[i] < t) t = timer[i];
        }
        return(t);      /* devuelve el valor del temporizador más bajo */
}


void print_statistics(filedesc fd, statistics st)
/* Imprime las estadísticas.
 */
{
        int word[3];                            /* almacena los datos que se le pasarán al main() */
        float mean_delay, mrtd;                 /* para los cálculos */
        bigint uso;                                     /* uso del enlace */
        float cad;                                      /* cadencia eficaz */
        float rend;                                     /* rendimiento */
        float ttotal;                           /* tiempo total de simulación */

        /* pasa el tiempo total de la simulación de ticks a segundos */
```

```
        ttotal = (float) st.ttotal / (float) st.tickseg;

        cad = (float) (st.bytes_trx * 8) / ttotal;          /* calcula la cadencia eficaz */
        rend = (cad / (float) st.rb) * 100;                 /* calcula el rendimiento */
        read(fd.cr, &(uso), sizeof(bigint));                /* lee los ticks que el transmisor ha estado
                                                                                                ocupado */

        st.uso = (float) uso * 100 / (float) st.ttotal;     /* calcula el uso del enlace */
        mean_delay = (float) st.delay / (float) st.delay_counted;   /* calcula el retardo medio */
        mrtd = (float) st.round_delay / (float) st.trip_cuonted;    /* retardo medio de vuelta */

        sleep(1);       /* espera a que el otro nodo imprima sus datos */
        printf("\nNodo %d :\n\n", fd.id);
        printf("\tEnviadas.\n");          /* estadísticas de transmisión */
        printf("\tTramas datos totales :   %9d\n", st.data_sent);
        printf("\tTramas retransmitidas :  %9d\n", st.data_retransmitted);
        printf("\tTramas ack :             %9d\n\n", st.acks_sent);

        printf("\tRecibidas.\n");         /* estadísticas de recepción */
        printf("\tTramas Ack sin error :   %9d\n", st.good_acks_recd);
        printf("\tTramas Ack con error :   %9d\n", st.cksum_acks_recd);
        printf("\tTramas datos sin error : %9d\n", st.good_data_recd);
        printf("\tTramas datos sin error : %9d\n", st.cksum_data_recd);
        printf("\tPaquetes entregados :    %9d\n\n", st.payloads_accepted);

        printf("\tVencimientos temporizadores.\n");    /* estadísticas de los temporizadores */
        printf("\tRetransmisión :          %9d\n", st.timeouts);
        printf("\tAsentimiento :           %9d\n\n", st.ack_timeouts);

        printf("\tEnlace.\n");            /* estadísticas del enlace */
        printf("\tRetardo medio :          %9.2f ms.\n", mean_delay);
        printf("\tRMD :                    %9.2f ms.\n", mrtd);
        printf("\tUso del enlace :         %9.2f %%\n", st.uso);
        printf("\tRendimiento :            %9.2f %%\n", rend);
        printf("\tCadencia eficaz :        %9.2f (bits/seg)\n", cad);
        fflush(stdin);

        word[0] = 0;                        /* el 0 indica fin de estadísticas */
        word[1] = st.payloads_accepted;     /* paquetes entregados al nivel de red */
        word[2] = st.data_sent;             /* tramas transmitidas */
        write(fd.mw, word, 3*sizeof(int));  /* manda datos al main() y le dice que ya ha
                                                                        imprimido sus datos */
        sleep(1);
        exit(0);
}

void sim_error(char *s, int fd)
/* Ha ocurrido un error durante la simulación.
 */
{
        bigint zero = 0;        /* señal de fin de la simulación */

        printf("%s\n", s);      /* imprime un mensaje relacionado con el error */
        write(fd, &zero, TICK_SIZE);        /* manda la señal de fin de la simulación al canal */
        exit(1);                            /* sale con error */
}

void manda_bandera(filedesc fd, statistics st)
/* manda la bandera de control que indica que en esta iteración no se van a
 * mandar más tramas.
 */
{
        int got;                    /* para comprobar que no hay error al mandar la bander */
        frame bandera;              /* trama que hace de bandera */

        bandera.kind = end;     /* para que se sepa que es la bandera */
        got = write(fd.aux, &bandera, FRAME_SIZE);          /* manda la bandera */
        if (got != FRAME_SIZE) print_statistics(fd, st);    /* error al mandar la bandera */
        return;
}

statistics inicia_estadisticas(void)
/* Inicializa las variables que se usarán para obtener datos estadisticos
 * del funcionamiento de los protocolos
 */
{
        statistics st;          /* estructura donde almacenar las estadísticas */

        st.data_sent = 0;                   /* tramas enviadas */
        st.data_retransmitted = 0;          /* tramas retransmitidas */
        st.acks_sent = 0;                   /* asentimientos enviados */

        st.good_data_recd = 0;              /* tramas de datos recibidas sin error */
        st.cksum_data_recd = 0;             /* tramas de datos recibidas con error */
        st.good_acks_recd = 0;              /* asentimientos recibidos sin error */
        st.cksum_acks_recd = 0;             /* asentimientos recibidos con error */
        st.payloads_accepted = 0;           /* paquetes entregados al nivel de red en secuencia */

        st.timeouts = 0;                    /* expiraciones de los temporizadores de retransmisión */
        st.ack_timeouts = 0;                /* expiraciones del temporizador de asentimiento */

        st.delay = 0;                       /* retardo */
        st.delay_counted = 0;               /* numero de tramas de las que se ha contado el retardo */
        st.round_delay = 0;                 /* retardo de vuelta */
        st.trip_cuonted = 0;                /* numero de tramas de las que se ha contado el retardo */
        st.bytes_trx = 0;                   /* número de bytes transmitidos */

        return(st);             /* devuelve la estructura con las variables inicializadas */
}

void manda_final(filedesc fd, statistics st)
{
        bigint word = END_FILE;
        if (write(fd.mw, &word, TICK_SIZE) != TICK_SIZE) print_statistics(fd, st);
}

nodo *enlista(frame fr, nodo *list)
{
/* Añade un nodo al principio de la lista que hace de buffer de transmisión
 */
        nodo *nuevo;

        nuevo = (nodo *) malloc(sizeof(nodo));      /* crea un nuevo nodo */
```

```
        if(nuevo == NULL){    /* compruba que había memoria para otro nodo */
                printf("ERROR .- No se pueden crear más nodos.\n");
                exit(1);
        }

        /* enlaza el nuevo nodo al principio de la lista */
        if(list != NULL) list->ant = nuevo;
        nuevo->sig = list;
        nuevo->ant = NULL;
        nuevo->fr = fr;                 /* completa el nuevo nodo */
        list = nuevo;           /* coloca el principio de la lista */
        return(list);
}

void comprueba_lista(buffer *b, timers *t, filedesc fd, statistics st, bigint tout)
/* Comprueba si el transmisor está ocupado. Si está libre, mira si hay tramas pendientes
 * (en la lista) y les actializa el campo ack y las transmite (las manda al transmisor).
 */
{
        bigint word = NOTHING;      /* almacena la respuesta del trasmisor */
        nodo *ppio = b->list;   /* puntero al principio de la lista */
        nodo *ult = b->list;    /* puntero al final de la lista */
        nodo *penult;                   /* puntero al nodo penúltimo de la lista */

        /* si el transmisor no está ocupado y hay nodos en la lista, manda una trama */
        /* transmisor libre: b->trans = true */
        if((b->trans) && (b->list != NULL)) {

                /* busca el último nodo */
                while(ult->sig != NULL) ult = ult->sig;
                penult = ult->ant;

                /* calcula de nuevo el campo ack según los valores actuales */
                ult->fr.ack = (b->frm_exp + b->max_seq) % (b->max_seq + 1);

                /* envía la nueva trama e inicia su temporizador */
                if (write(fd.ww, &(ult->fr), FRAME_SIZE) != FRAME_SIZE) print_statistics(fd, st);
                if(ult->fr.kind = data) start_timer(ult->fr.seq , tout, t);

                /* borra el último nodo de la lista */
                if(penult != NULL) penult->sig = NULL;
                else ppio = NULL;
                free(ult);
        }

        /* apunta la lista al primer nodo, si es que hay alguno */
        b->list = ppio;

        /* lee el estado del transmisor */
        if (read(fd.cr, &word, TICK_SIZE) != TICK_SIZE) print_statistics(fd, st);
        if (word == OK) b->trans = true;
        else b->trans = false;

        return;
}

void itoa(int num, char *cad, int base)
/* Pasa el número num a una cadena cad con la base base.
 */
{
        int aux;                /* variable auxiliar */
        char micad[10]; /* cadena intermedia */
        char micar[2];  /* carácter que se va añadiendo a la cadena */
        int i= 0;               /* variable índice */
        int j = 0;              /* variable índice */

        /* mientras el número no sea menor que la base... */
        while (num >= base){
                /* calcula el módulo del num / base y el resultado le suma 48 para pasarlo a ASCII */
                micad[i++] = 48 + (num % base);
                /* divide num por base y lo almacena el num */
                num /= base;
        }

        /* el residuo que queda también lo introduce en la cadena */
        micad[i] = 48 + num;

        /* le da la vuelta a la cadena, porque está al revés */
        while (i>=0) {
                /* va pasando los caracteres de micad a micar, desde el final al principio */
                micar[0] = micad[i--];
                /* termina micar con el terminador */
                micar[1] = '  \0'  ;
                /* añade este último caracter a la cadena que es parámetro de salida */
                strcat(cad, micar);
        }
        return;
}

void print_traza(traza_kind tipo, bigint tick, int id, frame f, seq_nr seq, statistics st)
/* Imprime la traza.
 * La cadena que se imprime depende del tipo de traza, del número de nodo, de los datos
 * de la trama y de las estadísticas almacenadas hasta el momento.
 * En el programa original la traza se limitaba a ser un simple mensaje que mostraba datos del
 * transcurso del programa. Ahora además de mostrar datos se prende dar una información
 * "gráfica" del tipo de traza que se muestra y de que hilo la produce.
 */
{
        char info[50];          /* cadena con los datos que se van a imprimir */
        char num[10];           /* número pasado de número a cadena */
        char dibujo[19];        /* cadena que muestra gráficamente el tipo de traza */
        char msg[140];                  /* mensaje que se va a imprimir */
        char frm[30];           /* información de la traza */

        dibujo[0] = '  \0'  ;       /* inicialización de las cadenas */
        info[0] = '  \0'  ;
        frm[0] = '  \0'  ;
        msg[0] = '  \0'  ;
        num[0] = '  \0'  ;

        itoa(tick/DELTA, num, 10);    /* pasa el número de tick actual a una cadena */
        strcat(info, num);                      /* añade el tick a la información */
        strcat(info, ". ");                     /* añade un '  .'   tras la cadena actual */
```

```
/* según el tipo de traza se incluye en ella distinta información */
/* se crean las cadenas dibujo e info, que después se colocarán debidamente en orden en
   la cadena msg, según el nodo que produce la traza */
switch(tipo){
        case periodic:  /* traza periódica */
                strcat(dibujo, "   | |XXXX| |   ");        /* dibujo de este tipo de traza */
                /* en esta traza se muestran las tramas enviadas, paquetes entregados, y número
                   de veces que han expirado los temporizadores */
                strcat(info, "-");
                num[0] = '  \0'  ;
                itoa(st.data_sent, num, 10);
                strcat(info, num);                                              /* añade las tramas enviadas */
                strcat(info, ", ");
                num[0] = '  \0'  ;
                itoa(st.payloads_accepted, num, 10);  /* añade los paquetes entregados */
                strcat(info, num);
                strcat(info, ", ");
                num[0] = '  \0'  ;
                itoa(st.timeouts, num, 10);                             /* añade los temporizadores */
                strcat(info, num);
                strcat(info, "-");                                              /* todo se añade a la cadena info */
                break;
        case tmout:
                strcat(dibujo, "   | |   | |   ");   /* dibujo de este tipo de traza */
                /* en esta traza se muestra el número de trama que produce el timeout */
                strcat(info, "frame ");             /* añade la palabra "frame" */
                num[0] = '  \0'  ;
                itoa(seq, num, 10);                     /* añade el número de trama */
                strcat(info, num);
                strcat(info, " timeout");           /* finaliza con "timeout" */
                break;
        case ack_tmout:
                strcat(dibujo, "   | |   | |   ");   /* dibujo de este tipo de traza */
                strcat(info, "ack timeout");        /* en esta traza solo se muestra "ack_timeout" */
                break;
        case send_pkt:
                /* en esta traza el dibujo depende del nodo que genera la traza y del número del
                   paquete que lleva la traza, ya que si es 0, indica comienzo o final del archivo */
                if(pktnum(f.info) == 0)
                        strcat(dibujo, (id == 0 ? " >>| |   | |   " : "   | |   | |<< "));
                else strcat(dibujo, (id == 0 ? " ->| |   | |   " : "   | |   | |<- "));
                fr(f, info);        /* y en info se pone la información de la trama */
                break;
        case del_pkt:
                /* en esta traza el dibujo depende del nodo que genera la traza y del número del
                   paquete que lleva la traza, ya que si es 0, indica final del archivo */
                if(pktnum(f.info) == 0)
                        strcat(dibujo, (id == 0 ? " <F| |   | |   " : "   | |   | |F> "));
                else strcat(dibujo, (id == 0 ? " <-| |   | |   " : "   | |   | |-> "));
                strcat(info, "pkt=");     /* y en info va el número del paquete */
                num[0] = '  \0'  ;
                itoa(pktnum(f.info), num, 10);
                strcat(info, num);
                break;
        case del_ini:    /* entrega del primer paquete del fichero al nivel de red */
                /* en esta traza el dibujo depende del nodo que genera la traza */
                strcat(dibujo, (id == 0 ? " <<| |   | |   " : "   | |   | |>> "));
                strcat(info, "pkt=");     /* y en info va el número del paquete */
                num[0] = '  \0'  ;
                itoa(pktnum(f.info), num, 10);
                strcat(info, num);
                break;
        case good_rec:
                /* en esta traza el dibujo depende del nodo que genera la traza */
                strcat(dibujo, (id == 0 ? "   |<-|   | |   " : "   | |   |->|   "));
                fr(f, info);                            /* en info se pone la información de la traza */
                strcat(info, " dly=");    /* y también del retardo que ha sufrido */
                num[0] = '  \0'  ;
                itoa(seq, num, 10);
                strcat(info, num);
                break;
        case bad_rec:
                /* en esta traza el dibujo depende del nodo que genera la traza */
                strcat(dibujo, (id == 0 ? "   |<X|   | |   " : "   | |   |X>|   "));
                fr(f, info);                            /* en info se pone la información de la traza */
                strcat(info, " dly=");    /* y también del retardo que ha sufrido */
                num[0] = '  \0'  ;
                itoa(seq, num, 10);
                strcat(info, num);
                break;
        case trx:
                /* en esta traza el dibujo depende del nodo que genera la traza */
                strcat(dibujo, (id == 0 ? "   |->|   | |   " : "   | |   |<-|   "));
                fr(f, info);                            /* en info se pone la información de la traza */
                strcat(info, " t_tx=");   /* y también el tamaño del paquete que transporta la trama */
                num[0] = '  \0'  ;
                itoa(seq, num, 10);
                strcat(info, num);
                break;
}

/* según el número de nodo va primero el dibujo y después la info o al revés */
/* además se ponen los espacios necesarios para que los dibujos aparezcan en el
   centro de la pantalla, y sea más gráfica la información mostrada */
switch(id){
        case 0:
                strcat(msg, info);                      /* primero la info */
                strcat(msg, dibujo);    /* y después el dibujo */
                printf("%57s\n", msg);          /* se añaden los espacios necesarios */
                break;
        case 1:
                strcat(msg, dibujo);    /* primero el dibujo */
                strcat(msg, info);                      /* y después la info */
                printf("%39s%s\n", " ", msg);  /* se añaden los espacios necesarios */
                break;
}
return;
}
```

9 Canal.c

```
/* En este fichero se encuentra el código correspondiente a los canales.
 * Las tramas que llegan del nodo se van metiendo en una lista doblemente enlazada.
 * Los nodos de la lista se crean y eliminan dinamicamente.
 * El estado de estas tramas en la lista es transmitiéndose (solo una), propagándose,
 * o esperando a ser transmitida. En cada tick se recorre la lista y se comprueeba si
 * ha terminado de transmitirse la última trama o si no hay nada transmitiendose y
 * hay tramas esperando.
 */

#include <string.h>
#include "common.h"
#include "hilossec.h"

/* Prototipo de las funciones */
bigint sincroniza(int ww, int mr, nodo *trans, bigint tick, bigint uso);
nodo *nuevos(nodo *prim, int cr, int mw, bigint tick);
nodo *transmisor(nodo *trans, nodo *prim, param_canal args, bigint tick,
                                int id, bigint *uso);
nodo *ultimo(nodo *prim);
nodo *comprueba_ultimo(nodo *prim, nodo *trans, int cw, bigint tick);
unsigned int pktnum(packet p);
char *fr(frame f, char *frm);
void imprime_traza(bigint tick, int id, int debug, frame fr, int t_trx);
void print_traza(traza_kind tipo, bigint tick, int id, frame f, seq_nr seq,
                                statistics st);

void canal(filedesc fd, param_canal args)
{
/* Esta función es la principal del canal. La llamada a esta función la realiza main().
 * Ejecuta simpre el mismo bucle hasta que le llega el fin de la simulación, que se
 * traduce en un tick == 0
 */

        nodo *trans = NULL;  /* apunta a la trama que se está transmitiendo */
        nodo *prim = NULL;   /* apunta al primer nodo de la lista */
        nodo *ult = NULL;    /* apunta al ultimo nodo de la lista */
        bigint tick = 0;             /* almacena el tiempo */
        bigint uso = 0;                      /* almacena el numero de ticks que está en uso el transmisor */

        do {
                /* lee el tick del nodo y le manda su estado */
                tick = sincroniza(fd.mw, fd.mr, trans, tick, uso);
                /* mira si el nodo ha mandado tramas */
                prim = nuevos(prim, fd.wr, fd.aux, tick);
                /* si la lista no esta vacía hace las correspondientes comprobaciones */
                if(prim != NULL) {
                        /* mira como va el transmisor */
                        trans = transmisor(trans, prim, args, tick, fd.id, &uso);
                        /* comprueba si el ultimo nodo ya se ha terminado de propagar */
                        prim = comprueba_ultimo(prim, trans, fd.ww, tick);
                }
        /* comprueba que aún no se ha recibido la señal de fin de simulación */
        } while(tick>0);
        exit(0);            /* finaliza la ejecución sin error */
}

bigint sincroniza(int ww, int mr, nodo *trans, bigint tick, bigint uso)
{
/* Escribe el estado del transmisor al nodo, y después se lee el tick del nodo.
 * Si el tick es 0 es la señal de fin de simulación, entonces se manda el uso del transmisor
 * al nodo para los cálculos y se finaliza.
 */
        bigint word = NOTHING;      /* estado del transmisor NOTHING = ocupado, OK = libre */

        if(trans == NULL) word = OK;          /* si el transmisor está libre madamos un OK */
        /* pero también mandamos el OK si al siguiente tick termina de transmitirse la trama
         * y no hay ninguna trama esperando. Esto se hace para evitar tiempos muertos entre tramas.
         */
        else if ((trans->timer == tick + 1 * DELTA) && (trans->sig == NULL)) word = OK;
        if(write(ww, &word, TICK_SIZE) != TICK_SIZE) {                      /* escribe el estado del canal */
                printf("Canal no puede escribir respuesta.\n"); /* error al escribir el estado */
                exit(1);
        }
        if(read(mr, &tick, TICK_SIZE) != TICK_SIZE) {                /* lee tick del nodo */
                printf("Canal no puede leer tick de main.\n");  /* error al leer el tick */
                exit(1);
        }
        if(tick==0) {                                                       /* señal de fin de programa */
                write(ww, &uso, sizeof(bigint));        /* manda basura al nodo para que no siga esperando */
        }
        return(tick);     /* devuelve el tick para que lo utilicen las demas funciones */
}

nodo *nuevos(nodo *prim, int cr, int mw, bigint tick)
{
/* Mira si el nodo ha pasado frames nuevos para ponerlos en cola.
 * En el programa original se utilizaba la función fstat() que devolvía el estado de un pipe,
 * en particular su tamaño, para comprobar si se había escrito algo. Pero esta función aunque
 * está incluída en las librerías de Cygwin, no funciona, por lo que se ha recurrido a un
 * truquillo para comprobar si se ha escrito algo en el pipe.
 * Lo que se hace es, antes de leer el pipe se introduce en él una trama de tipo end,
 * que es la bandera; después se lee el pipe hasta leer la bandera, que es el final del pipe.
 * Esto se hace así porque si se lee de un pipe vacío el programa se queda colgado esperando.
 */
        frame fr;                    /* para almacenar lo que mande el nodo */
        nodo *nuevo;   /* para apuntar al nuevo nodo si se crea */

        fr.kind = end;               /* identifica a la bandera */
        if(write(mw, &fr, FRAME_SIZE) != FRAME_SIZE) {                    /* introduce una bandera en el pipe */
                printf("Canal no puede escribir bandera.\n");  /* para saber cuando llega el final */
                exit(1);
        }
        /* lee del pipe hasta llegar a la bandera y procesa las tramas que llegen */
        do {
                if(read(cr, &fr, FRAME_SIZE) != FRAME_SIZE) {  /* lee una trama del pipe */
                        printf("Canal no puede escribir respuesta.\n");
                        exit(1);
                }
                if(fr.kind != end) {                /* no es la bandera, es una trama de verdad */
                        nuevo = (nodo *) malloc(sizeof(nodo));          /* crea un nuevo nodo */
                        if(prim == NULL) {                /* si la lista está vacía, pone la trama */
```

```
                        prim = nuevo;           /* en el primer nodo */
                        prim->sig = NULL;
                } else {                                /* lista no vacía */
                        prim->ant = nuevo;      /* se enlaza el nuevo nodo al principio*/
                        nuevo->sig = prim;
                        prim = nuevo;
                }
                prim->timer = 0;                /* esto indica que está esperando a ser tramsmitida */
                prim->fr = fr;                  /* introduce el nuevo frame en el nuevo nodo */
                prim->ant = NULL;               /* esto indica que es el promer nodo */
        }
    } while (fr.kind != end);
    return(prim);   /* devuelve la posición donde comienza la lista */
}


nodo *transmisor(nodo *trans, nodo *prim, param_canal args, bigint tick,
                                int id, bigint *uso)
{
/* Gestiona el transmisor.
 * Si hay alguna trama en el transmisor (trans != NULL) mira cuando termina de transmitirse,
 * en el dato trans->timer.
 * Si le toca a una trama terminar de transmitirse y pasar a propagarse, pone el tick
 * en que termina su propagación en trans->timer.
 * Si no se está transmitiendo ninguna trama se coge la siguiente de la lista y se calcula
 * cuando ha de terminar su transmisión
 */
    int t_trx;          /* para realizar calculos intermedios */
    if(trans != NULL) {             /* Hay una trama transmitiendose */
        if(trans->timer <= tick) {              /* Ha terminado de transmitirse */
            /* Pasa a propagarse, calcula el numero de ticks necesarios */
            t_trx = (strlen(trans->fr.info.linea) + 8) * 8 * args.ticks_seg;
            t_trx = (float) t_trx / (float) args.rb;
            /* se incrementa el uso de línea solo al terminar de transmitir */
            *uso += t_trx;
            /* se indica en el nodo cuando termina de transmitirse la trama */
            trans->timer = tick + args.t_prop;
            /* pasa a transmitir la la siguiente trama, si hay alguna */
            trans = trans->ant;
            if(trans != NULL) {             /* hay una trama esperando */
                /* calcula el tiempo que tardará en transmitirse */
                t_trx = (strlen(trans->fr.info.linea) + 8) * 8 * args.ticks_seg;
                t_trx = (float) t_trx / (float) args.rb;
                trans->timer = tick + t_trx * DELTA;
                /* imprime la traza correspondiente */
                imprime_traza(tick, id, args.debug, trans->fr, t_trx);
            }
        }
    } else if(prim->timer == 0){    /* no se está transmitiendo y hay tramas a la espera */
        trans = prim;                                   /* PROBLEMA SI LLEGAN VARIAS A LA VEZ */
        /* calcula el tiempo que tardará en transmitirse */
        t_trx = (strlen(trans->fr.info.linea) + 8) * 8 * args.ticks_seg;
        t_trx = (float) t_trx / (float) args.rb;
        trans->timer = tick + t_trx * DELTA;
        /* imprime la traza correspondiente */
        imprime_traza(tick, id, args.debug, trans->fr, t_trx);
    }
    return(trans);  /* devuelve un puntero al nodo que se está transmitiendo o NULL */
}

nodo *ultimo(nodo *prim)
{
/* Recorre la lista hasta encontrar el últmo nodo.
 * El último nodo se reconoce porque su puntero al siguiente nodo está a NULL.
 */
    nodo *aux = prim;       /* puntero auxiliar para recorrer la lista */

    while(aux->sig != NULL) aux = aux->sig;
    return(aux);    /* devuelve un puntero al último nodo */
}

nodo *comprueba_ultimo(nodo *prim, nodo *trans, int cw, bigint tick)
{
/* Comprueba si el último nodo se ha propagado ya y puede pasarse al al otro nodo.
 */
    nodo *aux;      /* puntero auxiliar para reordenar la lista */
    nodo *ult;      /* puntero al último nodo */

    ult = ultimo(prim); /* busca el último nodo */

    /* si el ultimo no se está transmitiendo, es que se está propagando, estonces si su
     * dato timer es menor que el tick quiere decir que ha terminado de propagarse
     */
    if(ult != trans && ult->timer <= tick)  {       /* ha terminado de propagarse */
        if(write(cw, &ult->fr, FRAME_SIZE) != FRAME_SIZE) {   /* se pasa la trama al otro nodo */
            printf("Canal no puede escribir tramas.");          /* error al escribir la trama */
            exit(1);
        }
        if(ult->ant != NULL) {          /* el penúltimo nodo pasa a ser el último, y por eso */
            aux = ult->ant;             /* debe tener sig a NULL */
            aux->sig = NULL;
        } else prim = NULL;
        free(ult);                              /* libera la memoria del último nodo */
    }
    return(prim); /* devuelve un puntero al primer nodo de la lista */
}

void imprime_traza(bigint tick, int id, int debug, frame f, int t_trx)
{
/* Imprime la traza correspondiente si así lo ha pedido el usuario en el parámetro traza.
 * La imprime en el descriptor de fichero t_trx que puede ser un fichero o la salida estandar.
 */
    statistics st;

    if(debug & TRANSMITS) print_traza(trx, tick, id, f, t_trx, st);
}
```

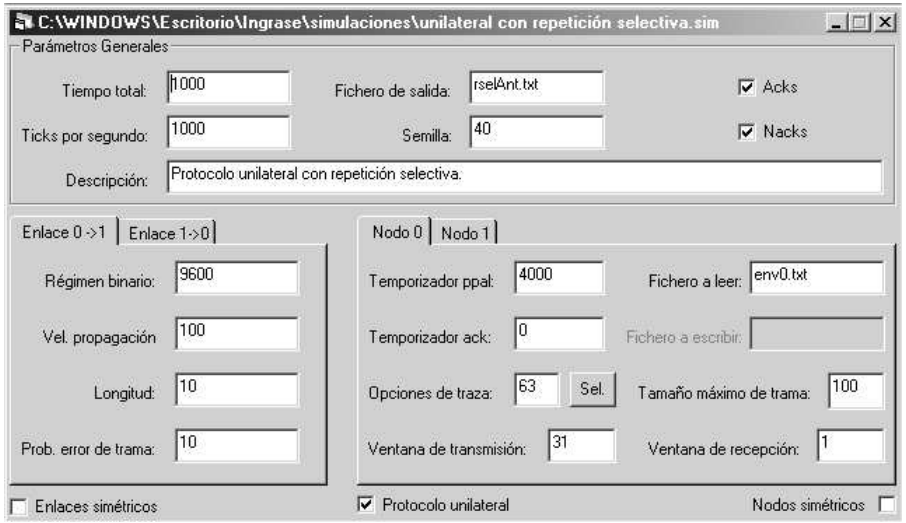
Capítulo 9. Código de InGraSE

1 Formulario principal *frmMain*.


```
'  InGraSE v.1.00
'  Interfaz gráfica del Simulador de potocolos de nivel de Enlace
'  Desarrollado por Juan Luis Millán Romera
Option Explicit

'  Const EM_UNDO = &HC7

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal wMsg As
Long, ByVal wParam As Long, ByVal lParam As Any) As Long
Private Declare Function OSWinHelp% Lib "user32" Alias "WinHelpA" (ByVal hWnd&, ByVal HelpFile$, ByVal
wCommand%, dwData As Any)

'  directorio donde se encuentra el simulador sim.exe
Public SimDir As String
'  almacena una cadena que expresa el parámetro sobre el que se realiza la batería
Public mBateria As String
Dim mButton As Button

Private Sub MDIForm_Load()
    '  Lee del registro del sistema las posiciones de la ventana la última vez
    '  que se cerró
    Me.Left = GetSetting(App.Title, "Settings", "MainLeft", 1000)
    Me.Top = GetSetting(App.Title, "Settings", "MainTop", 1000)
    Me.Width = GetSetting(App.Title, "Settings", "MainWidth", 6500)
    Me.Height = GetSetting(App.Title, "Settings", "MainHeight", 6500)

    '  Deshabilita los botones innecesarios
    tbToolBar.Buttons(3).Enabled = False
    tbToolBar.Buttons(5).Enabled = False
    tbToolBar.Buttons(6).Enabled = False
    tbToolBar.Buttons(7).Enabled = False
    tbToolBar.Buttons(9).Enabled = False
    tbToolBar.Buttons(10).Enabled = False

    '  redimensionas las matrices dinámicas y las inicializa
    ReDim Document(0)
    ReDim FState(0)
    ReDim Sims(0)

    '  el directorio de simulación es el directorio actual
    SimDir = CurDir
    mBateria = ""

End Sub

Private Sub MDIForm_Unload(Cancel As Integer)
    '  Escribe en el registro del sistema la posicion de la ventana
    '  Si no está minimizada
    If Me.WindowState <> vbMinimized Then
        SaveSetting App.Title, "Settings", "MainLeft", Me.Left
        SaveSetting App.Title, "Settings", "MainTop", Me.Top
        SaveSetting App.Title, "Settings", "MainWidth", Me.Width
        SaveSetting App.Title, "Settings", "MainHeight", Me.Height
    End If
End Sub

Private Sub tbToolBar_ButtonClick(ByVal Button As MSComctlLib.Button)
    '  Establece la acción a realizar según el boton que se pulse de la
    '  barra de herramientas
    '  Algunas acciones son funciones de este formulario y otras del
    '  formulario secundario activo
    On Error Resume Next
    Select Case Button.Key
        Case "Nuevo"
            mnuFileNew_Click
        Case "Abrir"
            mnuFileOpen_Click
        Case "Guardar"
            ActiveForm.mnuFileSave_Click
        Case "Cortar"
            ActiveForm.mnuEditCut_Click
        Case "Copiar"
            ActiveForm.mnuEditCopy_Click
        Case "Pegar"
            ActiveForm.mnuEditPaste_Click
        Case "Simular"
            ActiveForm.mnuToolsSim_Click
        Case "Resultados"
            ActiveForm.mnuResult_Click
    End Select
End Sub

Public Sub mnuHelpAbout_Click()
    '  Muestra la ventana Acerca de
    frmAbout1.Show vbModal, Me
```

```
End Sub

Private Sub mnuHelpContents_Click()
   Dim nRet As Integer
   ' si no hay archivo de ayuda para este proyecto, mostra un mensaje al usuario
   ' puede establecer el archivo de Ayuda para la aplicación en el cuadro
   ' de diálogo Propiedades del proyecto
   If Len(App.HelpFile) = 0 Then
      MsgBox "No se puede mostrar el contenido de la Ayuda. No hay Ayuda asociada a este proyecto.", vbInformation,
Me.Caption
   Else
      On Error Resume Next
      '  Error al abrir el fichero de ayuda
      nRet = OSWinHelp(Me.hWnd, App.HelpFile, 3, 0)
      If Err Then
         MsgBox Err.Description
      End If
   End If

End Sub


Private Sub mnuToolsOptions_Click()
   ' Muestra el formulario Opciones
   ' Para un uso futuro
   ' frmOptions.Show vbModal, Me
End Sub

Private Sub mnuViewStatusBar_Click()
   ' Muestra u oculta la línea de estado
   mnuViewStatusBar.Checked = Not mnuViewStatusBar.Checked
   sbStatusBar.Visible = mnuViewStatusBar.Checked
End Sub

Private Sub mnuViewToolBar_Click()
   ' Muestra u oculta la barra de herramientas
   mnuViewToolbar.Checked = Not mnuViewToolbar.Checked
   tbToolBar.Visible = mnuViewToolbar.Checked
End Sub


Public Sub mnuFileExit_Click()
   ' descargar el formulario
   Unload Me
End Sub

Public Sub mnuFileOpen_Click()
   ' Muestra el cuadro de diálogo Abrir archivo
   Dim sFile As String
   With dlgCommonDialog
   .FileName = ""
      ' Establece los indicadores y los atributos del control common dialog
      .DialogTitle = "Abrir"
      .CancelError = False
      .Filter = "Archivos InGraSE (*.sim)|*.sim|Todos los archivos (*.*)|*.*"
      ' Abre la ventana Abrir archivo
      .ShowOpen
      If Len(.FileName) = 0 Then
         ' Si no se selecciona ningún archivo se sale
         Exit Sub
      End If
      ' Almacena en la variable sFile la ruta y el nombre del archivo seleccionado
      sFile = .FileName
   End With
   ' Abre el archivo seleccionado
   OpenDoc sFile

End Sub

Public Sub mnuFileNew_Click()
   ' Crea un archivo nuevo
   LoadNewDoc
End Sub
```

2 Formulario secundario *frmDocument.*




```
Option Explicit

'  Constantes y funciones necesaras para realizar la llamada al simulador
'  y esperar a que la simulación termine
Private Const PROCESS_QUERY_INFORMATION = &H400
Private Const STILL_ACTIVE = &H103
Private Declare Function OpenProcess Lib "kernel32" _
   (ByVal dwDesiredAccess&, ByVal bInheritHandle&, ByVal dwProcessId&) _
   As Long
```

```
Private Declare Function GetExitCodeProcess Lib "kernel32" _
   (ByVal hProcess As Long, lpExitCode As Long) _
   As Long

'   objeto Simulación que contiene los datos, estado y resultados de la simulación
Public mSim As New Simulacion
'   directorio donde se ha cargado el fichero de simulación
Dim MiDir As String

Sub Borra(Fich As String)
   '   Borra el fichero Fich, que se encuentra en el directorio MiDir
   '   para unir el directorio con el nombre del fichero, la cadena del directorio
   '   ha de estar terminada por '  \'
   If Right(MiDir, 1) <> "\" Then
      '   si se produce algun error se continua la ejecución
      '   por ejemplo, puede no existir el fichero
      On Error Resume Next
      Kill MiDir & "\" & Fich
   Else
      On Error Resume Next
      Kill MiDir & Fich
   End If
End Sub

Private Sub CompruebaNumeros(KeyAscii As Integer, Control As TextBox)
   '   Impide que en el TextBox Control que se pasa como parámetro se
   '   introduzca ningún caracter que no sea un número
   If KeyAscii = 13 Then
      KeyAscii = 0        '  Para que no "pite"
      SendKeys "{tab}"    '  Envia una pulsación TAB
   ElseIf KeyAscii <> 8 Then   '  El 8 es la tecla de borrar (backspace)
   '  Si después de añadirle la tecla actual no es un número...
      If Not IsNumeric("0" & Control.Text & Chr(KeyAscii)) Then
      '  ... se desecha esa tecla y se avisa de que no es correcta
         Beep
         KeyAscii = 0
      End If
      '   al modificar un parámetro hay que rehacer la simulación
      mSim.Realizada = False
      frmMain.tbToolBar.Buttons(10).Enabled = False
   End If
End Sub

Public Sub GeneraParam()
   '   pasa los datos del formulario al objeto Simulación
   '   nombre del fichero de entrada a sim.exe
   mSim.Result.Archivo = tbFil
   '   parámetros generales
   With mSim.Param.General
      .Descr = Me.tbDesc
      .Ttotal = Me.tbTtotal
      .TickSeg = Me.tbTickseg
      .FichSal = Me.tbFil
      .Semilla = Me.tbSemilla
      If Me.chkAck Then .Acks = True Else .Acks = False
      If Me.chkNack Then .Nacks = True Else .Nacks = False
   End With
   '   parámetros del canal 0
   With mSim.Param.Canal0
      .Rb = Me.tbRb(0)
      .Vp = Me.tbVp(0)
      .Lon = Me.tbLong(0)
      .Prob = Me.tbError(0)
   End With
   '   parámetros del canal 1
   With mSim.Param.Canal1
      .Rb = Me.tbRb(1)
      .Vp = Me.tbVp(1)
      .Lon = Me.tbLong(1)
      .Prob = Me.tbError(1)
   End With
   '   parámetros del nodo 0
   With mSim.Param.Nodo0
      .TempPpal = Me.tbTemp(0)
      .TempAck = Me.tbAckTemp(0)
      .Traza = Me.tbTraza(0)
      .FichLeer = Me.tbFichLeer(0)
      .FichEsc = Me.tbFichEsc(0)
      .Mtu = Me.tbMtu(0)
      .Vtrx = Me.tbVtrx(0)
      .Vrx = Me.tbVrx(0)
   End With
   '   parámetros del nodo 1
   With mSim.Param.Nodo1
      .TempPpal = Me.tbTemp(1)
      .TempAck = Me.tbAckTemp(1)
      .Traza = Me.tbTraza(1)
      .FichLeer = Me.tbFichLeer(1)
      .FichEsc = Me.tbFichEsc(1)
      .Mtu = Me.tbMtu(1)
      .Vtrx = Me.tbVtrx(1)
      .Vrx = Me.tbVrx(1)
   End With
End Sub

Public Sub Cambia_Traza(Index As Integer, Traza As Integer)
   tbTraza(Index) = Traza
End Sub

Public Sub PresentaParam()
   '   rellena el formulario con los datos contenidos en un objeto Simulación
   With mSim.Param
      '   parámetros generales
      With .General
         tbTtotal = .Ttotal
         tbDesc = .Descr
         tbFil = .FichSal
         tbSemilla = .Semilla
         tbTickseg = .TickSeg
         If .Acks Then chkAck.Value = 1 Else chkAck.Value = 0
         If .Nacks Then chkNack.Value = 1 Else chkNack.Value = 0
      End With
      '   parámetros del canal 0
      With .Canal0
```

```
        tbRb(0) = .Rb
        tbVp(0) = .Vp
        tbLong(0) = .Lon
        tbError(0) = .Prob
    End With
    ' parámetros del canal 1
    With .Canal1
        tbRb(1) = .Rb
        tbVp(1) = .Vp
        tbLong(1) = .Lon
        tbError(1) = .Prob
    End With
    ' parámetros del nodo 0
    With .Nodo0
        tbTemp(0) = .TempPpal
        tbAckTemp(0) = .TempAck
        tbTraza(0) = .Traza
        tbFichLeer(0) = .FichLeer
        tbFichEsc(0) = .FichEsc
        tbVtrx(0) = .Vtrx
        tbVrx(0) = .Vrx
        tbMtu(0) = .Mtu
    End With
    ' parámetros del nodo 1
    With .Nodo1
        tbTemp(1) = .TempPpal
        tbAckTemp(1) = .TempAck
        tbTraza(1) = .Traza
        tbFichLeer(1) = .FichLeer
        tbFichEsc(1) = .FichEsc
        tbVtrx(1) = .Vtrx
        tbVrx(1) = .Vrx
        tbMtu(1) = .Mtu
    End With
End With
' comprueba si los canales son simétricos y si es así verifica la
' casilla correspondiente
If tbRb(0) = tbRb(1) Then
    If tbVp(0) = tbVp(1) Then
        If tbLong(0) = tbLong(1) Then
            If tbError(0) = tbError(1) Then
                chkSimCan.Value = 1
            Else
                chkSimCan.Value = 0
            End If
        End If
    End If
End If
' comprueba si los nodos son simétricos y si es así verifica la
' casilla correspondiente
If tbTemp(0) = tbTemp(1) And tbAckTemp(0) = tbAckTemp(1) And _
    tbTraza(0) = tbTraza(1) And tbVtrx(0) = tbVtrx(1) _
    And tbVrx(0) = tbVrx(1) Then
    chkSimNod.Value = 1
Else
    chkSimNod.Value = 0
End If
    ' comprueba si se define un protocolo unilateral
    ' para ello el fichero a escribir del nodo 0 y el fichero a leer del nodo 1
    ' han de estar vacíos
    If tbFichEsc(0) = "" And tbFichLeer(1) = "" Then chkProt.Value = 1
    ' selecciona las pestañas que se van a ver cuando se presente el formulario
    SSTabCanal.Tab = 0
    SSTabNodo.Tab = 0
End Sub

Private Sub chkProt_Click()
    Dim Habilitado As Boolean
    ' Según esté marcada o no esta casilla, habilita o deshabilita
    ' los TextBox correspondientes para generar un protocolo unilateral
    ' En Habilirado se guarda el estado actual
    Habilitado = Not CBool(chkProt.Value)
    chkProt.Tag = chkProt.Value
    If Not Habilitado Then
        tbFichEsc(0) = ""
        lblFichEsc(0).Enabled = False
        tbFichEsc(0).Enabled = False
        ' tb se cambia el color para mejorar el efecto visual
        tbFichEsc(0).BackColor = &H80000004
        tbFichLeer(1) = ""
        lblFichLeer(1).Enabled = False
        tbFichLeer(1).Enabled = False
        tbFichLeer(1).BackColor = &H80000004
    Else
        tbFichEsc(0).Enabled = True
        lblFichEsc(0).Enabled = True
        tbFichEsc(0).BackColor = &H80000005
        lblFichLeer(1).Enabled = True
        tbFichLeer(1).Enabled = True
        tbFichLeer(1).BackColor = &H80000005
    End If
End Sub

Private Sub Form_Activate()
    ' este evento se ejecuta al pasar de un formulario secundario a otro
    ' Habilita los botones necesarios al activar el formulario
    frmMain.tbToolBar.Buttons(3).Enabled = True
    frmMain.tbToolBar.Buttons(5).Enabled = True
    frmMain.tbToolBar.Buttons(6).Enabled = True
    frmMain.tbToolBar.Buttons(7).Enabled = True
    frmMain.tbToolBar.Buttons(9).Enabled = True
    ' habilita el boton de ver resultados según se haya realizado o no
    ' la simulación
    frmMain.tbToolBar.Buttons(10).Enabled = mSim.Realizada

End Sub

Private Sub Form_Deactivate()
    ' este evento se ejecuta al pasar de un formulario secundario a otro
    ' Deshabilita los botones innecesarios
    frmMain.tbToolBar.Buttons(3).Enabled = False
    frmMain.tbToolBar.Buttons(5).Enabled = False
    frmMain.tbToolBar.Buttons(6).Enabled = False
    frmMain.tbToolBar.Buttons(7).Enabled = False
```

```
    frmMain.tbToolBar.Buttons(9).Enabled = False
    frmMain.tbToolBar.Buttons(10).Enabled = False
End Sub


Private Sub Form_GotFocus()
    '   habilita el boton de ver resultados según se haya realizado o no
    '   la simulación cuando el formulario obtiene el foco
    frmMain.tbToolBar.Buttons(10).Enabled = mSim.Realizada
End Sub

Private Sub Form_Load()
    '   establece el tamaño del formulario
    Me.Width = 9645
    Me.Height = 5580
    '   Habilita los botones necesarios
    frmMain.tbToolBar.Buttons(3).Enabled = True
    frmMain.tbToolBar.Buttons(5).Enabled = True
    frmMain.tbToolBar.Buttons(6).Enabled = True
    frmMain.tbToolBar.Buttons(7).Enabled = True
    frmMain.tbToolBar.Buttons(9).Enabled = True
    '   habilita el boton de ver resultados según se haya realizado o no
    '   la simulación
    frmMain.tbToolBar.Buttons(10).Enabled = mSim.Realizada
    '   almacena en MiDir el directorio desde donde se ha cargado el último fichero
    '   CurDir() es una fución del sistema
    MiDir = CurDir
End Sub


Private Sub Form_Resize()
    '   cuando se cambia el tamaño del formulario, evita que se establezca
    '   a un tamaño mayor del debido
    If Me.WindowState = vbNormal Then
        If Me.Width > 9645 Then Me.Width = 9645
        If Me.Height > 5580 Then Me.Height = 5580
    End If
End Sub


Private Sub chkAck_GotFocus()
    '   muestra mensajes de información en la línea de estado
    frmMain.sbStatusBar.Panels(1) = "Indica si pueden usarse tramas de asentimiento sin datos."
End Sub


Private Sub chkNack_GotFocus()
    '   muestra mensajes de información en la línea de estado
    frmMain.sbStatusBar.Panels(1) = "Indica si pueden usarse tramas de asentimiento negativo."
End Sub


Private Sub chkSimCan_Click()
    Dim Habilitado As Boolean
    '   Según se marque o no esta casilla, habilita o deshabilita
    '   los TextBox correspondientes para generar un protocolo con canales simétricos
    '   En Habilirado se guarda el estado actual
    Habilitado = Not CBool(chkSimCan.Value)
    chkSimCan.Tag = chkSimCan.Value
    '   si se marca la casilla, se copian los valores del canal 0 al canal 1
    If Not Habilitado Then
        tbRb(1) = tbRb(0)
        tbVp(1) = tbVp(0)
        tbLong(1) = tbLong(0)
        tbError(1) = tbError(0)
    End If
    '   se habilitan o deshabilitan los controles correspondientes
    lblRb(1).Enabled = Habilitado
    tbRb(1).Enabled = Habilitado
    lblVp(1).Enabled = Habilitado
    tbVp(1).Enabled = Habilitado
    lblLong(1).Enabled = Habilitado
    tbLong(1).Enabled = Habilitado
    lblError(1).Enabled = Habilitado
    tbError(1).Enabled = Habilitado
End Sub

Private Sub chkSimNod_Click()
    Dim Habilitado As Boolean
    '   Según se marque o no esta casilla, habilita o deshabilita
    '   los TextBox correspondientes para generar un protocolo con nodos simétricos
    '   En Habilirado se guarda el estado actual
    Habilitado = Not CBool(chkSimNod.Value)
    chkSimNod.Tag = chkSimNod.Value
    '   si se marca la casilla, se copian los valores del nodo 0 al nodo 1
    If Not Habilitado Then
        tbTemp(1) = tbTemp(0)
        tbAckTemp(1) = tbAckTemp(0)
        tbTraza(1) = tbTraza(0)
        tbVtrx(1) = tbVtrx(0)
        tbVrx(1) = tbVrx(0)
        tbMtu(1) = tbMtu(0)
    End If
    '   se habilitan o deshabilitan los controles correspondientes
    lblTemp(1).Enabled = Habilitado
    tbTemp(1).Enabled = Habilitado
    lblAckTemp(1).Enabled = Habilitado
    tbAckTemp(1).Enabled = Habilitado
    lblTraza(1).Enabled = Habilitado
    tbTraza(1).Enabled = Habilitado
    lblVtrx(1).Enabled = Habilitado
    tbVtrx(1).Enabled = Habilitado
    lblVrx(1).Enabled = Habilitado
    tbVrx(1).Enabled = Habilitado
    cmdTraza(1).Enabled = Habilitado
    tbMtu(1).Enabled = Habilitado
    lblMtu(1).Enabled = Habilitado
End Sub

Private Sub cmdTraza_Click(Index As Integer)
    '   abre la ventana de traza
    '   si el TextBox de traza está vacío, pone un 0 para evitar errores después
    If tbTraza(Index) = "" Then tbTraza(Index) = "0"
    '   rellena los parámetros necesarios para abrir la ventana traza
```

```
    frmTraza.Traza = tbTraza(Index)
    frmTraza.Index = Index
    '   muestra la ventana traza
    frmTraza.Show 1
End Sub

Private Sub cmdTraza_GotFocus(Index As Integer)
    '   muestra mensajes de información en la línea de estado
    frmMain.sbStatusBar.Panels(1) = "Seleccionar las opciones de traza."
End Sub


Private Sub Form_Unload(Cancel As Integer)
    '   Muestra la instancia del formulario actual como eliminada
    FState(Me.Tag).Deleted = True
    mSim.Realizada = False
    '   libera las referencias al objeto Simulación para que se elimine
    Set mSim = Nothing
    Form_Deactivate
End Sub

Private Sub mniWindowCascade_Click()
    '   pone en cascada todas las ventanas de los formularios secundarios
    frmMain.Arrange vbCascade
End Sub

Private Sub mniWindowTileHorizontal_Click()
    '   ordena horizontalmente todas las ventanas de los formularios secundarios
    frmMain.Arrange vbTileHorizontal
End Sub

Private Sub mnuEditCopy_Click()
    '   tareas futuras
    On Error Resume Next
    '  Clipboard.SetText frmMain.ActiveForm.ActiveControl
End Sub

Public Sub mnuEditCut_Click()
    '   tareas futuras
    On Error Resume Next
    '  Clipboard.SetText ActiveForm.rtfText.SelRTF
    '  ActiveForm.rtfText.SelText = vbNullString
End Sub

Public Sub mnuEditPaste_Click()
    '   tareas futuras
    On Error Resume Next
    '  ActiveForm.rtfText.SelRTF = Clipboard.GetText
End Sub

Public Sub mnuEditUndo_Click()
    '  TareasPendientes: Agregar código '  mnuEditUndo_Click'  .
    MsgBox "Agregar código '  mnuEditUndo_Click'  ."
End Sub

Private Sub mnuFileClose_Click()
    '   cierra la ventana activa actual
    Unload Me
End Sub

Private Sub mnuFileExit_Click()
    '   ejecuta la misma función en el formulario principal
    frmMain.mnuFileExit_Click
End Sub

Private Sub mnuFileNew_Click()
    '   ejecuta la misma función en el formulario principal
    frmMain.mnuFileNew_Click
End Sub

Private Sub mnuFileOpen_Click()
    '   ejecuta la misma función en el formulario principal
    frmMain.mnuFileOpen_Click
End Sub

Public Sub mnuFileSave_Click()
    '   guarda el formulario en un archivo de simulación
    '   strFilename almacena la ruta y el nombre del fichero que se crea
    Dim strFilename As String

    If Left(Me.Caption, 8) = "Document" Then
        '   Aún no se ha guardado el archivo.
        '   Obtiene el nombre del archivo y después llama al procedimiento de guardar.
        strFilename = ""
    Else
        '   El título del formulario contiene el nombre del archivo abierto.
        strFilename = Me.Caption
    End If

    '   Llama al procedimiento de guardar. Si Filename está vacía, es porque el
    '   usuario eligió Cancelar en el cuadro de diálogo Guardar como; si no,
    '   guarda el archivo.
    If strFilename <> "" Then
        '   establece el parámetro Param.Archivo del objeto mSim con la ruta
        '   y el nombre del archivo donde se van a escribir sus datos
        mSim.Param.Archivo = strFilename
        '   pasa los datos del formulario al objeto mSim
        GeneraParam
        '   escribe el objeto mSim en un archivo
        mSim.Param.Escribir
    Else
        '   si no hay nombre del fichero, se llama a la función mnuFileSaveAs
        mnuFileSaveAs_Click
    End If

End Sub

Private Sub mnuFileSaveAs_Click()
    '   abre la ventana Guardar Como...
    '   en FileName se guarda la ruta y el nombre con el que se va a crear
    '   el nuevo fichero de simulación
    Dim FileName As String

    With frmMain.dlgCommonDialog
```

```
        ' establece las propiedades del cuado de diálogo guardar como
        .FileName = ""
        .DialogTitle = "Guardar como..."
        .CancelError = False
        ' establece los indicadores y los atributos del control common dialog
        .Filter = "Archivos InGraSE (*.sim)|*.sim|Todos los archivos (*.*)|*.*"
        ' abre el control common dialog como guardar como
        .ShowSave
        ' si no se ha escrito el nombre, no se guarda nada
        If Len(.FileName) = 0 Then
            Exit Sub
        End If
        ' en la propiedad del common dialog .FileName se devuelve el nombre
        ' del fichero que se ha escrito o elegido
        FileName = .FileName
    End With

    ' si hay un nombre de archivo se crea éste
    If FileName <> "" Then
        ' establece el parámetro Param.Archivo del objeto mSim con la ruta
        ' y el nombre del archivo donde se van a escribir sus datos
        Me.mSim.Param.Archivo = FileName
        ' pasa los datos del formulario al objeto mSim
        GeneraParam
        ' escribe el objeto mSim en un archivo
        mSim.Param.Escribir
        ' cambia el título del formulario activo
        Me.Caption = FileName
    End If

End Sub

Private Sub mnuHelpAbout_Click()
    ' abre el formulario Acerca de
    frmMain.mnuHelpAbout_Click
End Sub

Private Sub mnuHelpContents_Click()
    Dim nRet As Integer

    ' si no hay archivo de ayuda para este proyecto, mostrar un mensaje al usuario
    ' puede establecer el archivo de Ayuda para la aplicación en el cuadro
    ' de diálogo Propiedades del proyecto
    If Len(App.HelpFile) = 0 Then
        MsgBox "No se puede mostrar el contenido de la Ayuda. No hay Ayuda asociada a este proyecto.", vbInformation,
Me.Caption
    Else
        ' On Error Resume Next
        ' nRet = OSWinHelp(Me.hWnd, App.HelpFile, 3, 0)
        ' If Err Then
        '     MsgBox Err.Description
        ' End If
    End If
End Sub

Public Sub mnuResult_Click()
    ' presenta los resultados de la simulación si esta se ha realizado
    Dim frmResNew As frmRes

    If mSim.Realizada Then
        ' crea una nueva instancia del objeto frmRes
        Set frmResNew = New frmRes
        ' rellena el formulario creado con los datos de mSim
        frmResNew.Mostrar mSim
        ' muestra el formulario frmResNew
        frmResNew.Show 1
    End If
End Sub

Private Sub mnuToolsBateria_Click()
    ' Muestra el formulario frmBat
    frmBat.Show 0
End Sub

Public Sub CrearGrafica(EjeX As String)
    ' Crea una gráfica con los resultados de las simulaciones de la batería
    ' en principio los datos que se muestran son los del rendimiento en el eje Y
    ' y los del eje X dependerán del parámetro que se ha variado en la batería

    ' crea una nueva instancia del objeto frmResSim
    Dim frmRS As New frmResSim
    ' variables índice
    Dim i As Integer
    Dim j As Integer
    Dim Tot As Integer

    ' inicializa variables
    j = 0

    ' si no se ha pasado ningún parámetro, se supone probabilidad de error
    If EjeX = "" Then EjeX = "Probabilidad de error"

    ' comprueba que todas las simulaciones estén hechas
    For i = 1 To UBound(Module1.Document)
        If Module1.FState(i).Deleted <> True And _
            Module1.Document(i).mSim.Realizada = False Then
            MsgBox "Se han de realizar todas las simulaciones antes de crear una gráfica", vbExclamation
            Exit Sub
        End If
    Next

    ' fija los títulos de los ejes y el nombre de la gráfica
    frmRS.EjeX = EjeX
    ' muestra el formulario con la gráfica
    frmRS.Show 1

End Sub
Private Sub mnuToolsGraf_Click()
    ' llama a la función CrearGráfica pasandole como parámetro
    ' el parámetro sobre el que se ha realizado la batería
    CrearGrafica frmMain.mBateria
End Sub

Private Sub mnuToolsOptions_Click()
```

```
    '   para un futuro desarrollo
    '   frmOptions.Show vbModal, Me
End Sub


Public Sub mnuToolsSim_Click()
    '   arrancanca una nueva simulación sobre el formulario secundario activo

    '   cambia la información mostrada en la línea de estado
    frmMain.sbStatusBar.Panels(1) = "Simulando. Espera un poco..."

    '   la función Simular() comienza la simulación y devuelve una valor booleano
    '   según haya sido satisfactoria o no la simulación
    If Simular Then
        MsgBox "La simulación ha terminado."
        '   habilita el botón correspondiente en la barra de botones
        frmMain.tbToolBar.Buttons(10).Enabled = True
        '   cambia la información mostrada en la línea de estado
        frmMain.sbStatusBar.Panels(1) = "Simulación finalizada."
        '   muestra los resultados
        mnuResult_Click
    Else
        '   se ha producido un error durante la simulación
        MsgBox "Se ha producido un error durante la simulación."
    End If
End Sub

Function Simular() As Boolean
    '   realiza una simulación y devuelve un valor booleano con el resultado

    '   variables necesarias para detectar cuando termina la simulación
    Dim hShell As Long
    Dim hProc As Long
    Dim codExit As Long
    '   para almacenar la orden que se pasa al sistema operativo
    Dim miOrden As String

Inicio:
    '   los datos del formulario se guardan en un archivo temporal
    mSim.Param.Archivo = "temp.sim"
    GeneraParam
    mSim.Param.Escribir

    '   Borra el fichero de resultados si existe
    Borra mSim.Param.General.FichSal

    '   Prepara la llamada al simulador
    '   cambia al directorio donde se encuentra el fichero de simulación
    ChDir MiDir
    '   crea la orden que pasa al sistema operativo
    miOrden = frmMain.SimDir & "\sim.exe temp.sim"
    '   si se produce algún error ve a la etiqueta ErrorHandler:
    On Error GoTo ErrorHandler:
    '   hace la llamada al sistema en una ventana minimizada y sin el foco
    '   en hShell se recoge el número de proceso para luego monitorizarlo
    hShell = Shell(miOrden, vbMinimizedNoFocus)
    '   borra el registro de errores
    On Error GoTo 0

    '   espera a que se complete el proceso
    hProc = OpenProcess(PROCESS_QUERY_INFORMATION, False, hShell)
    Do
        GetExitCodeProcess hProc, codExit
        DoEvents
    Loop While codExit = STILL_ACTIVE

    '   se ha terminado la simulación y se pasa a leer los resultados
    frmMain.sbStatusBar.Panels(1) = "Leyendo los resultados..."

    '   la función Leer() devuelve en boolean con el resultado de la lectura
    If mSim.Result.Leer Then
        '   lectura correcta
        mSim.Realizada = True
        '   borra el archivo temp.sim
        If Right(MiDir, 1) <> "\" Then
            Kill MiDir & "\temp.sim"
        Else
            Kill MiDir & "temp.sim"
        End If
        '   establece a true el parámetro a devolver indicando que no ha
        '   habido ningún error
        Simular = True
    Else
        '   error en la lectura
        MsgBox "Error al leer los resultados." & vbCrLf _
            & "Comprueba que no falten los ficheros que leer y" & vbCrLf _
            & "que estén escritos los nombres de los ficheros a escribir."
        '   cambia la información mostrada en la línea de estado
        frmMain.sbStatusBar.Panels(1) = "Error."
        '   establece a False el parámetro a devolver indicando que si ha
        '   habido error
        Simular = False
    End If
    GoTo Fin:

ErrorHandler:
    '   se produjo un error durante la simulación, se muestra un mensaje
    MsgBox ("No se encuentra el simulador sim.exe")
    '   establece a False el parámetro a devolver indicando que si ha
    '   habido error
    Simular = False
    '   por si el problema es que nos se ha encontrado el simulador, se abre
    '   un common dialog para que el usuario de su posición
    With frmMain.dlgCommonDialog
        '   nombre a buscar
        .FileName = "sim.exe"
        .DialogTitle = "Buscar Simulador"
        .CancelError = False
        '   establece los indicadores y los atributos del control common dialog
        .Filter = "Simulador (sim.exe)|sim.exe|Todos los archivos (*.*)|*.*"
        '   muestra la nueva ventana
        .ShowOpen
        '   si no se ha expeificado la nueva ubicación sale de la función
        If Len(.FileName) = 0 Then
```

```
        GoTo Fin:
      End If
      '   cambia la ruta a la nueva
      frmMain.SimDir = CurDir
      '   vuelve a intentar simular
      GoTo Inicio:
    End With

Fin:
End Function

Private Sub mnuViewRefresh_Click()
    '  TareasPendientes: Agregar código '  mnuViewRefresh_Click'  .
    MsgBox "Agregar código '  mnuViewRefresh_Click'  ."
End Sub

Private Sub mnuViewStatusBar_Click()
    '   Muestra u oculta la línea de estado
    mnuViewStatusBar.Checked = Not mnuViewStatusBar.Checked
    frmMain.sbStatusBar.Visible = mnuViewStatusBar.Checked
End Sub

Private Sub mnuViewToolBar_Click()
    '   Muestra u oculta la barra de botones
    mnuViewToolBar.Checked = Not mnuViewToolBar.Checked
    frmMain.tbToolBar.Visible = mnuViewToolBar.Checked
End Sub

Private Sub mnuWindowArrangeIcons_Click()
    '   Organiza los iconos de las ventanas minimizadas
    frmMain.Arrange vbArrangeIcons
End Sub

Private Sub mnuWindowTileVertical_Click()
    '   Ordena verticalmente las vantans de los formularios secundarios
    frmMain.Arrange vbTileVertical
End Sub

Private Sub tbAckTemp_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
    frmMain.sbStatusBar.Panels(1) = "Temporizador para los asentimientos, en ms."
End Sub

Private Sub tbAckTemp_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
    '   index es el indice para diferenciar entre los dos nodos
    '   KeyAscii es la última tecla pulsada
    Call CompruebaNumeros(KeyAscii, tbAckTemp(Index))
End Sub

Private Sub tbAckTemp_LostFocus(Index As Integer)
    '   si los nodos son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del nodo 1
    If Me.chkSimNod.Value And Index = 0 Then tbAckTemp(1) = tbAckTemp(0)
End Sub

Private Sub tbAckTemp_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero temporizador de asentimiento
    '   al poner cancel a True se impide que se salga del TextBox
    If Val(tbAckTemp(Index)) < 0 Then
        MsgBox "Este valor no puede ser negativo", vbOKOnly, "Temporizador de asentimiento"
        Cancel = True
    End If
End Sub

Private Sub tbDesc_GotFocus()
    '   cambia el contenido de la línea de estado al entrar en este parámetro
    frmMain.sbStatusBar.Panels(1) = "Descripción de la simulación."
End Sub

Private Sub tbEntrada_GotFocus()
    '   cambia el contenido de la línea de estado al entrar en este parámetro
    frmMain.sbStatusBar.Panels(1) = "Esto no debería de mostrarse."
End Sub

Private Sub tbError_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
    frmMain.sbStatusBar.Panels(1) = "Probabilidad de eror de trama, en %."
End Sub

Private Sub tbError_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
    Call CompruebaNumeros(KeyAscii, tbError(Index))
End Sub

Private Sub tbError_LostFocus(Index As Integer)
    '   si los canales son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del canal 1
    If chkSimCan.Value And Index = 0 Then tbError(1) = tbError(0)
End Sub

Private Sub tbError_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero probabilidad de error de trama
    If Val(tbError(Index)) < 0 Or Val(tbError(Index)) > 100 Then
        MsgBox "Este valor ha ser entero y estar entre 0 y 100", vbOKOnly, "Probabilidad de error de trama"
        Cancel = True
    End If
End Sub

Private Sub tbFichEsc_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
    frmMain.sbStatusBar.Panels(1) = "Fichero donde escribir el fichero recibido."
End Sub

Private Sub tbFichLeer_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
    frmMain.sbStatusBar.Panels(1) = "Fichero de donde leer el fichero a enviar."
End Sub

Private Sub tbFil_GotFocus()
    '   cambia el contenido de la línea de estado al entrar en este parámetro
    frmMain.sbStatusBar.Panels(1) = "Fichero donde guardar los resultados de la simulación."
End Sub
```

```
Private Sub tbLong_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
   frmMain.sbStatusBar.Panels(1) = "Longitud de la línea, en metros."
End Sub

Private Sub tbLong_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbLong(Index))
End Sub

Private Sub tbLong_LostFocus(Index As Integer)
    '   si los canales son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del canal 1
   If chkSimCan.Value And Index = 0 Then tbLong(1) = tbLong(0)
End Sub

Private Sub tbLong_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero longitud del enlace
   If Val(tbLong(Index)) < 1 Then
      MsgBox "Este valor ha ser mayor que 0", vbOKOnly, "Longitud del enlace"
      Cancel = True
   End If
End Sub

Private Sub tbMtu_Change(Index As Integer)
    '   si los nodos son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del nodo 1
   If chkSimNod.Value And Index = 0 Then tbMtu(1) = tbMtu(0)
End Sub

Private Sub tbMtu_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
   frmMain.sbStatusBar.Panels(1) = "Tamaño máximo de trama, en octetos. Se utiliza una cabecera de 8 octetos."
End Sub

Private Sub tbMtu_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbMtu(Index))
End Sub

Private Sub tbMtu_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero tamaño máximo de trama
   If Val(tbMtu(Index)) < 8 Or Val(tbMtu(Index)) > 100 Then
      MsgBox "Este valor ha de estar entre 8 y 100", vbOKOnly, "Tamaño máximo de trama"
      Cancel = True
   End If
End Sub

Private Sub tbRb_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
   frmMain.sbStatusBar.Panels(1) = "Régimen binario de la línea, en bits/s."
End Sub

Private Sub tbRb_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbRb(Index))
End Sub

Private Sub tbRb_LostFocus(Index As Integer)
    '   si los canales son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del canal 1
   If chkSimCan.Value And Index = 0 Then tbRb(1) = tbRb(0)
End Sub

Private Sub tbRb_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero régimen binario
   If Val(tbRb(Index)) < 1 Then
      MsgBox "Este valor ha ser mayor que 0 y 100", vbOKOnly, "Régimen binario"
      Cancel = True
   End If
End Sub

Private Sub tbSemilla_GotFocus()
    '   cambia el contenido de la línea de estado al entrar en este parámetro
   frmMain.sbStatusBar.Panels(1) = "Semilla utilizada para generar números pseudo-aleatorios."
End Sub

Private Sub tbSemilla_KeyPress(KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbSemilla)
End Sub

Private Sub tbSemilla_Validate(Cancel As Boolean)
    '   Valida el valor del parámero semilla
   If Val(tbTtotal) < 0 Then
      MsgBox "Este valor ha ser positivo", vbOKOnly, "Semilla"
      Cancel = True
   End If
End Sub

Private Sub tbTemp_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
   frmMain.sbStatusBar.Panels(1) = "Temporizador para la retransmisión, en ms."
End Sub

Private Sub tbTemp_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbTemp(Index))
End Sub

Private Sub tbTemp_LostFocus(Index As Integer)
    '   si los canales son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del canal 1
   If chkSimCan.Value And Index = 0 Then tbTemp(1) = tbTemp(0)
End Sub

Private Sub tbTemp_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero temporizador de retransmisiones
   If Val(tbTemp(Index)) < 1 Then
      MsgBox "Este valor ha de ser mayor de 0", vbOKOnly, "Temporizador de retransmisiones"
      Cancel = True
   End If

End Sub
```

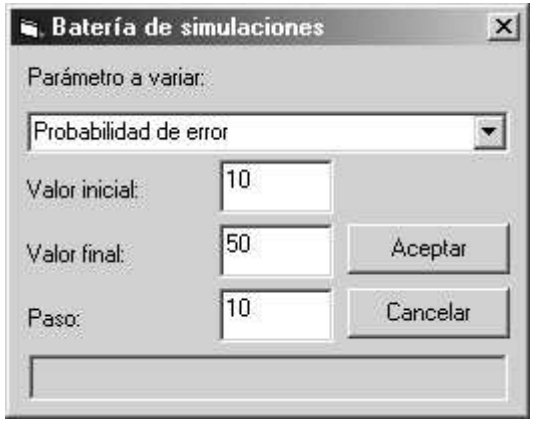
```
Private Sub tbTickseg_GotFocus()
    '   cambia el contenido de la línea de estado al entrar en este parámetro
   frmMain.sbStatusBar.Panels(1) = "Número de ticks por segundo. A más ticks mayor precisión."
End Sub

Private Sub tbTickseg_KeyPress(KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbTickseg)
End Sub

Private Sub tbTickseg_Validate(Cancel As Boolean)
    '   Valida el valor del parámero ticks por segundo
   If Val(tbTickseg) < 1 Then
      MsgBox "Este valor ha ser mayor que 0", vbOKOnly, "Ticks por segundo"
      Cancel = True
   End If
End Sub

Private Sub tbTraza_Change(Index As Integer)
    '   si los nodos son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del nodo 1
   If chkSimNod.Value And Index = 0 Then tbTraza(1) = tbTraza(0)
End Sub

Private Sub tbTraza_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
   frmMain.sbStatusBar.Panels(1) = "Mensejes de depuración que muestra el simulador."
End Sub

Private Sub tbTraza_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbTraza(Index))
End Sub

Private Sub tbTraza_LostFocus(Index As Integer)
    '   si los nodos son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del nodo 1
   If chkSimNod.Value And Index = 0 Then tbTraza(1) = tbTraza(0)
End Sub

Private Sub tbTraza_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero traza
   If Val(tbTraza(Index)) < 0 Or Val(tbTraza(Index)) > 63 Then
      MsgBox "Este valor ha de estar entre 0 y 63", vbOKOnly, "Opciones de traza"
      Cancel = True
   End If

End Sub

Private Sub tbTtotal_GotFocus()
    '   cambia el contenido de la línea de estado al entrar en este parámetro
    frmMain.sbStatusBar.Panels(1) = "Tiempo total de simulación en segundos."
End Sub

Private Sub tbTtotal_KeyPress(KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbTtotal)
End Sub

Private Sub tbTtotal_Validate(Cancel As Boolean)
    '   Valida el valor del parámero tiempo total de simulación
   If Val(tbTtotal) < 1 Then
      MsgBox "Este valor ha ser mayor que 0", vbOKOnly, "Tiempo total"
      Cancel = True
   End If
End Sub

Private Sub tbVp_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
   frmMain.sbStatusBar.Panels(1) = "Velocidad de propagación de la línea, en m/s."
End Sub

Private Sub tbVp_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbVp(Index))
End Sub

Private Sub tbVp_LostFocus(Index As Integer)
    '   si los canales son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del canal 1
   If chkSimCan.Value And Index = 0 Then tbVp(1) = tbVp(0)
End Sub

Private Sub tbVp_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero ventana de propagación
   If Val(tbVp(Index)) < 1 Then
      MsgBox "Este valor ha ser mayor que 0", vbOKOnly, "Velocidad de propagación"
      Cancel = True
   End If
End Sub

Private Sub tbVrx_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
   frmMain.sbStatusBar.Panels(1) = "Tamaño de la ventana de recepción."
End Sub


Private Sub tbVrx_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
   Call CompruebaNumeros(KeyAscii, tbVrx(Index))
End Sub

Private Sub tbVrx_LostFocus(Index As Integer)
    '   si los nodos son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del nodo 1
   If chkSimNod.Value And Index = 0 Then tbVrx(1) = tbVrx(0)
End Sub

Private Sub tbVrx_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero ventana de recepción
   If Val(tbVrx(Index)) < 1 Then
      MsgBox "Este valor ha ser mayor que 0", vbOKOnly, "Ventana de recepción"
      Cancel = True
```

```
    End If
End Sub


Private Sub tbVtrx_GotFocus(Index As Integer)
    '   cambia el contenido de la línea de estado al entrar en este parámetro
    frmMain.sbStatusBar.Panels(1) = "Tamaño de la ventana de transmisión."
End Sub

Private Sub tbVtrx_KeyPress(Index As Integer, KeyAscii As Integer)
    '   comprueba que en este parámetro solo se introducen números
    Call CompruebaNumeros(KeyAscii, tbVtrx(Index))
End Sub

Private Sub tbVtrx_LostFocus(Index As Integer)
    '   si los nodos son simétricos copia el valor de este parámatro
    '   en su correspondiente parámetro del nodo 1
    If chkSimNod.Value And Index = 0 Then tbVtrx(1) = tbVtrx(0)
End Sub

Private Sub tbVtrx_Validate(Index As Integer, Cancel As Boolean)
    '   Valida el valor del parámero ventana de transmisión
    If Val(tbVtrx(Index)) < 1 Then
        MsgBox "Este valor ha ser mayor que 0", vbOKOnly, "Ventana de transmisión"
        Cancel = True
    End If
End Sub
```

3 Formulario Opciones de traza *frmTraza.*


```
Option Explicit

'   Valores constantes para calcular el valor final del parámetro traza
Const TRANSMITS = &H1   '   tramas transmitidas
Const DELIVERS = &H2    '   paquetes entregados
Const SENDS = &H4       '   tramas enviadas
Const RECEIVES = &H8    '   tramas recividas
Const TIMEOUTS = &H10   '   timeouts
Const PERIODIC = &H20   '   periódico

'   valor de la traza
Public Traza As Integer
'   para diferenciar entre la traza del nodo 0 y del nodo 1
Public Index As Integer

Private Sub CancelButton_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    '   Establece qué checkboxes han de estar marcadas según el valor inicial
    '   de la traza
    If (Traza And TRANSMITS) Then chkTrans.Value = 1
    If (Traza And DELIVERS) Then chkEntr.Value = 1
    If (Traza And SENDS) Then chkEnv.Value = 1
    If (Traza And RECEIVES) Then chkRec.Value = 1
    If (Traza And TIMEOUTS) Then chkTim.Value = 1
    If (Traza And PERIODIC) Then chkPer.Value = 1
End Sub


Private Sub OKButton_Click()
    '   Calcula el nuevo valor de la traza según los campos marcados
    Dim MiTraza As Integer
    MiTraza = 0
    If chkTrans.Value Then MiTraza = MiTraza + TRANSMITS
    If chkEnv.Value Then MiTraza = MiTraza + SENDS
    If chkEntr.Value Then MiTraza = MiTraza + DELIVERS
    If chkRec.Value Then MiTraza = MiTraza + RECEIVES
    If chkTim.Value Then MiTraza = MiTraza + TIMEOUTS
    If chkPer.Value Then MiTraza = MiTraza + PERIODIC
    '   pone el nuevo valor calculado en el formulario de simulación activo
    frmMain.ActiveForm.Cambia_Traza Index, MiTraza
    '   descarga este formulario
    Unload Me
End Sub
```

4 Formulario Batería de simulaciones *frmBat.*

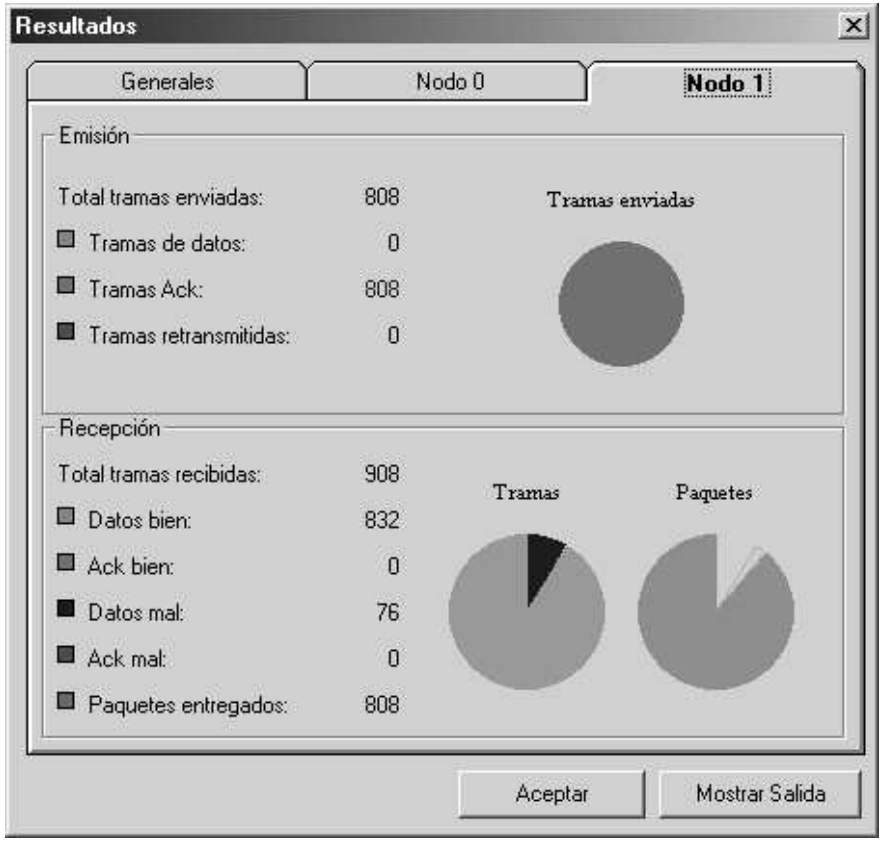



```
Option Explicit

'   Apunta al TextBox del formulario de simulación que contiene el valor actual
'   del parámetro sobre el que se va a realizar la batería
Dim mTB As TextBox
'   Indica si se ha pulsado el botón cancelar una vez comenzada la batería
Dim mCancelar As Boolean
'   Indica si ya ha comenzado la batería de simulaciones
Dim mSimulando As Boolean

Private Sub ComienzaBatería()
    '   Inicia el proceso de las simulaciones
    '   variables índice
    Dim i As Long
    Dim j As Long
    '   variables auxiliares
    Dim aux As Long
    Dim aux2 As Double
    Dim ctrl As Control
    Dim frmAct As frmDocument
    Set frmAct = frmMain.ActiveForm

    '   impide que se pueda lanzar otra batería mientras transcurre la actual
    OKButton.Enabled = False
    '   para que se pueda cancelar la batería sin esperar a que finalice
    mCancelar = False
    '   indica que ha comenzado la batería
    mSimulando = True
    '   en el formulario de la simulación actual, pone el valor del parámetro
    '   elegido para la batería al valor inicial
    mTB = tbVini
    '   inicializa los índices
    i = tbPaso
    j = tbVini
    j = j + i
    '   realiza los cálculos necesarios para la barra de progreso
    aux2 = tbVfin - tbVini
    aux2 = (aux2 / tbPaso) + 1
    aux2 = 100 / aux2
    aux = aux2

    '   comprueba si la simulacion del formulario actual ya se ha realizado y
    '   si no es así, se realiza
    With frmAct
        If Not .mSim.Realizada Then
            .Simular
            .mSim.Result.Leer
            .mSim.Realizada = True
        End If
    End With

    '   inicializa la barra de progreso
    Me.ProgressBar1.Value = aux

    For i = j To tbVfin Step tbPaso
        '   si se ha pulsado el botón cancelar se termina el bucle y la batería
        If mCancelar Then Exit For
        '   crea una copia del formilario de simulación actual con los mismos valores
        CreaCopia frmMain.ActiveForm
        Set frmAct = frmMain.ActiveForm
        '   muestra el formulario de batería para que no quede oculto bajo el
        '   nuevo formulario que se ha creado
        Me.Show
        '   cambia el valor del parámetro elegido en el nuevo formulario creado
        Select Case Combo1
        Case "Probabilidad de error"
            frmAct.tbError(0) = i
            If frmAct.chkSimCan Then frmAct.tbError(1) = i
        Case "Semilla"
            frmAct.tbSemilla = i
        Case "Ventana de recepción"
            frmAct.tbVrx(0) = i
            If frmAct.chkSimNod Then frmAct.tbVrx(1) = i
        Case "Ventana de transmision"
            frmAct.tbVtrx(0) = i
            If frmAct.chkSimNod Then frmAct.tbVtrx(1) = i
        Case "Temporizador"
            frmAct.tbTemp(0) = i
            If frmAct.chkSimNod Then frmAct.tbTemp(1) = i
        Case "Tamaño máx. de trama"
            frmAct.tbMtu(0) = i
            If frmAct.chkSimNod Then frmAct.tbMtu(1) = i
        Case "Longuitud del enlace"
            frmAct.tbLong(0) = i
            If frmAct.chkSimCan Then frmAct.tbLong(1) = i
        Case "Régimen binario"
            frmAct.tbRb(0) = i
            If frmAct.chkSimCan Then frmAct.tbRb(1) = i
        End Select
```

```
      ' comprueba que no se ha realizado la simulación y la hace
      With frmAct
        If Not .mSim.Realizada Then
          .Simular
          ' lee los resultados
          .mSim.Result.Leer
          .mSim.Realizada = True
          ' activa el botón de la barra de botones para indicar visualmente
          ' que se ha realizado la simulación de ese formulario
          frmMain.tbToolBar.Buttons(10).Enabled = True
        End If
      End With
      ' actualiza la barra de progreso
      ' si el valor nuevo supera el 100, se pone a 100
      If Me.ProgressBar1.Value + aux > 100 Then aux = 100 - Me.ProgressBar1.Value
      Me.ProgressBar1.Value = Me.ProgressBar1.Value + aux
    Next
    ' indica que ya no se esta simulando
    mSimulando = False
    ' inicia el proceso de creación de la gráfica
    frmAct.CrearGrafica Combo1
    ' descarga este formulario
    Unload Me
End Sub


Private Sub CompruebaNumeros(KeyAscii As Integer, Control As TextBox)
    ' Comprueba que en el TextBox que se pasa como entrada solo se
    ' puedan entroducir números
    ' KeyAscii es la última tecla pulsada
    If KeyAscii = 13 Then
        KeyAscii = 0        ' Para que no "pite"
        SendKeys "{tab}"    ' Envia una pulsación TAB
    ElseIf KeyAscii <> 8 Then   ' El 8 es la tecla de borrar (backspace)
    ' Si después de añadirle la tecla actual no es un número...
        If Not IsNumeric("0" & Control.Text & Chr(KeyAscii)) Then
        ' ... se desecha esa tecla y se avisa de que no es correcta
            Beep
            KeyAscii = 0
        End If
    End If
End Sub

Private Function ValidaPaso() As Boolean
    ' Comprueba que el valor introducido en el campo paso es un valor válido
    ' respecto al valor máximo y al mínimo
    Dim mInt As Integer
    ValidaPaso = False
    mInt = Val(tbVfin) - Val(tbVini)
    ' El paso no puede ser cero
    If Val(tbPaso) <> 0 Then
        ' el paso debe ser un divisor entero de la diferencia entre el
        ' valor máximo y el mínimo
        mInt = mInt Mod tbPaso
        If mInt <> 0 Then
            MsgBox "No es un paso correcto"
            ValidaPaso = True
        End If
    Else
        MsgBox "El paso no puede ser cero."
        ValidaPaso = True
    End If
End Function

Private Sub CancelButton_Click()
    ' Si se está simulando cuando se pulsa el botón cancelar se pone la variable
    ' mCancelar a true para que no se ejecute la siguiente simulación
    If mSimulando Then
        mCancelar = True
        ' se desactiva el boton Cancelar para indicar que se esta esperando a que
        ' termine la simulación en curso
        CancelButton.Enabled = False
    Else
    ' si no se estaba simulando simplemente se descarga el formulario
        Unload Me
    End If
End Sub


Private Sub Combo1_Click()
    ' Según el valor elegido en el control combo rellena los demás campos
    ' con los valores de la simulación activa
    Select Case Combo1
    Case "Longuitud del enlace"
        Set mTB = frmMain.ActiveForm.tbLong(0)
    Case "Semilla"
        Set mTB = frmMain.ActiveForm.tbSemilla
    Case "Tamaño máx. de trama"
        Set mTB = frmMain.ActiveForm.tbMtu(0)
    Case "Ventana de recepción"
        Set mTB = frmMain.ActiveForm.tbVrx(0)
    Case "Temporizador"
        Set mTB = frmMain.ActiveForm.tbTemp(0)
    Case "Probabilidad de error"
        Set mTB = frmMain.ActiveForm.tbError(0)
    Case "Ventana de transmision"
        Set mTB = frmMain.ActiveForm.tbVtrx(0)
    Case "Régimen binario"
        Set mTB = frmMain.ActiveForm.tbRb(0)
    Case Else
        MsgBox "No has seleccionado ningún campo correcto."
    End Select
    ' rellena los campos valor final y valor inicial, y paso
    Me.tbVini = mTB
    tbVfin = mTB
    tbPaso = "0"
    ' indica al formulario principal el tipo de batería realizada
    frmMain.mBateria = Combo1
End Sub


Private Sub Combo1_Validate(Cancel As Boolean)
    ' Evita que se produzca un error al intentar iniciar la batería sin
    ' haber seleccionado el tipo
```

```
   Select Case Combo1
   Case "Longuitud del enlace"
   Case "Semilla"
   Case "Tamaño máx. de trama"
   Case "Ventana de recepción"
   Case "Temporizador"
   Case "Probabilidad de error"
   Case "Ventana de transmision"
   Case "Régimen binario"
   '   si no se ha seleccionado nada todavía cancela el proceso
   Case Else
          MsgBox "No has seleccionado ningún campo correcto." & vbCrLf & "Selecciona un campo de la lista desplegable."
      Cancel = True
   End Select
End Sub


Private Sub Form_Load()
   '   al cargar el formulario da valor 0 al indicador de progreso
   '   para que no aparezcan valores ficticios
   Me.ProgressBar1.Value = 0
End Sub


Private Sub OKButton_Click()
   '   Hace las comprobaciones necesarias antes de iniciar la batería
   '   comprueba que se ha seleccionado un parámetro
   If Combo1 = "" Then
      MsgBox "Primero has de seleccionar el parametro a variar."
   Else
   '   compueba que el valor final es mayor que el inicial
      If Val(tbVfin) <= Val(tbVini) Then
         MsgBox "El valor final ha de ser mayor que el valor inicial"
      Else
      '   si no hay ninún problema comienza la batería
         If Not ValidaPaso Then ComienzaBatería
      End If
   End If
End Sub

Private Sub tbPaso_KeyPress(KeyAscii As Integer)
   '   Comprueba que los valores introducidos en el campo paso son números
   Call CompruebaNumeros(KeyAscii, tbPaso)
End Sub


Private Sub tbPaso_Validate(Cancel As Boolean)
   '   Valida el valor introducido en el campo paso
   Cancel = ValidaPaso
End Sub


Private Sub tbVfin_KeyPress(KeyAscii As Integer)
   '   Comprueba que los valores introducidos en el campo valor final son números
   Call CompruebaNumeros(KeyAscii, tbVfin)
End Sub


Private Sub tbVfin_Validate(Cancel As Boolean)
   '   Valida el valor introducido en el campo valor final
   If Val(tbVfin) <= Val(tbVini) Then
      MsgBox "El valor final ha de ser mayor que el valor inicial"
      Cancel = True
   End If
End Sub

Private Sub tbVini_KeyPress(KeyAscii As Integer)
   '   Valida el valor introducido en el campo valor inicial
   Call CompruebaNumeros(KeyAscii, tbVini)
End Sub
```

5 Formulario Resultados de la simulación *frmRes*.

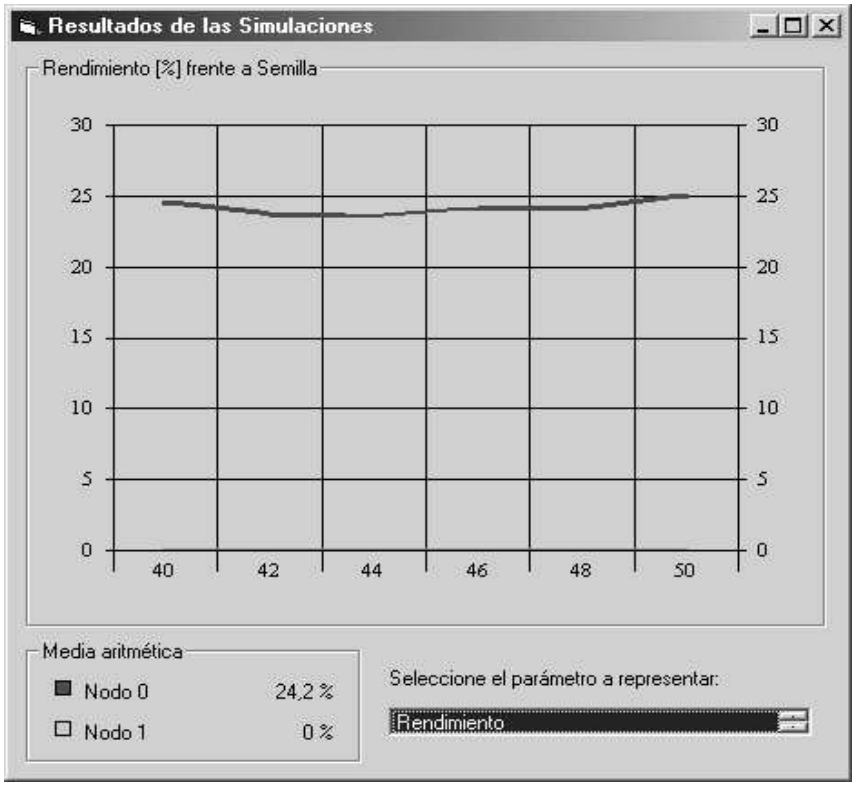



```
Option Explicit

Option Base 1

'   Simulación de la que se van a apresentar los resultados
Dim SimActual As Simulacion
'   Establece el modo de los gráficos entre barras apliladas o gráficos de sectores
Private Modo As Boolean
Public Sub Mostrar(Sim As Simulacion)
    '   Saca los valores del objeto Sim y los muestra en el formulario
    '   Al ser una función pública puede ser llamada desde fuera de este formulario
    Dim MiInt As Integer
    '   arrays de datos para dar valores a los objetos mschart
    Dim arrDataEnv0(1, 1 To 4)
    Dim arrDataEnv1(1, 1 To 4)
    Dim arrDataRec0(1 To 2, 1 To 8)
    Dim arrDataRec1(1 To 2, 1 To 8)
```

```
Set SimActual = Sim
With SimActual.Result
    ' Datos globales de la simulación
    lblTotal = .Ttotal & " ms."
    lblTsim = .Tsim & " ms."
    With .ResCan1
        ' Resultados del canal 1
        lblUso(1) = (.Uso / 100) & " %"
        lblRM(0) = (.MeanTrip / 100) & " ms."
        lblRMV(1) = (.MRoundTrip / 100) & " ms."
        lblCef(0) = (.Cef / 100) & " bits/s."
        lblRend(0) = (.Rend / 100) & " %"
        ' Rellena los datos de la barra de uso de la línea
        MiInt = .Uso / 100
        ' Selecciona el rango de valores de la barra
        pbUso(1).Max = 100
        pbUso(1).Min = 0
        pbUso(1).Value = MiInt
        ' Rellena los datos de la barra de rendimiento del enlace
        MiInt = .Rend / 100
        pbRend(0).Max = 100
        pbRend(0).Min = 0
        pbRend(0).Value = MiInt
    End With
    With .ResCan0
        ' Resultados del canal 0
        lblUso(0) = (.Uso / 100) & " %"
        lblRM(1) = (.MeanTrip / 100) & " ms."
        lblRMV(0) = (.MRoundTrip / 100) & " ms."
        lblCef(1) = (.Cef / 100) & " bits/s."
        lblRend(1) = (.Rend / 100) & " %"
        ' barra de uso de la línea
        MiInt = .Uso / 100
        pbUso(0).Max = 100
        pbUso(0).Min = 0
        pbUso(0).Value = MiInt
        ' barra de rendimiento
        MiInt = .Rend / 100
        pbRend(1).Max = 100
        pbRend(1).Min = 0
        pbRend(1).Value = MiInt
    End With
    With .ResNod0
        ' Resultados del nodo 0
        ' Presentación de los datos "simples"
        lblTotEnv(0) = .TotEnv
        lblDatEnv(0) = .DatosEnv
        lblAckEnv(0) = .AckEnv
        lblRet(0) = .Retr
        lblTotRec(0) = .TotRec
        lblDatOK(0) = .DatOK
        lblDatMal(0) = .DatMal
        lblAckOK(0) = .AckOK
        lblAckMal(0) = .AckMal
        lblPkt(0) = .Pkt
        ' Preparación de los datos para los mschart
        ' mschart de tramas enviadas
        ' primero damos nombre a los campos
        arrDataEnv0(1, 1) = "Tramas enviadas"
        ' y después le damos valores
        arrDataEnv0(1, 4) = .DatosEnv
        arrDataEnv0(1, 3) = .AckEnv
        arrDataEnv0(1, 2) = .Retr
        ' y por último los asignamos al mschart
        chtEnv(0).ChartData = arrDataEnv0
        ' mschart de tramas recibidas, tiene dos capas
        ' primero damos nombre a los campos
        arrDataRec0(1, 1) = "Tramas"
        arrDataRec0(2, 1) = "Paquetes"
        ' y después le damos valores
        ' los campos con valor 0 se utilizan como delimitadores
        ' para mejorar la presentación visual
        arrDataRec0(1, 7) = 0
        arrDataRec0(1, 6) = 0
        arrDataRec0(1, 5) = .DatOK
        arrDataRec0(1, 4) = .AckOK
        arrDataRec0(1, 3) = .DatMal
        arrDataRec0(1, 2) = .AckMal
        arrDataRec0(2, 8) = .Pkt
        arrDataRec0(2, 7) = .DatOK - .Pkt
        arrDataRec0(2, 6) = .TotRec - .DatOK
        arrDataRec0(2, 5) = 0
        arrDataRec0(2, 4) = 0
        arrDataRec0(2, 3) = 0
        arrDataRec0(2, 2) = 0
        arrDataRec0(2, 2) = 0
        ' y por último los asignamos al mschart
        chtRec(0).ChartData = arrDataRec0
    End With
    With .ResNod1
        ' Resultados del nodo 1
        ' Presentación de los datos "simples"
        lblTotEnv(1) = .TotEnv
        lblDatEnv(1) = .DatosEnv
        lblAckEnv(1) = .AckEnv
        lblRet(1) = .Retr
        lblTotRec(1) = .TotRec
        lblDatOK(1) = .DatOK
        lblDatMal(1) = .DatMal
        lblAckOK(1) = .AckOK
        lblAckMal(1) = .AckMal
        lblPkt(1) = .Pkt
        arrDataEnv1(1, 1) = "Tramas enviadas"
        arrDataEnv1(1, 4) = .DatosEnv
        arrDataEnv1(1, 3) = .AckEnv
        arrDataEnv1(1, 2) = .Retr
        chtEnv(1).ChartData = arrDataEnv1
        arrDataRec1(1, 1) = "Tramas"
        arrDataRec1(2, 1) = "Paquetes"
        arrDataRec1(1, 7) = 0
        arrDataRec1(1, 6) = 0
        arrDataRec1(1, 5) = .DatOK
        arrDataRec1(1, 4) = .AckOK
        arrDataRec1(1, 3) = .DatMal
```

```
        arrDataRec1(1, 2) = .AckMal
        arrDataRec1(2, 8) = .Pkt
        arrDataRec1(2, 7) = .DatOK - .Pkt
        arrDataRec1(2, 6) = .TotRec - .DatOK
        arrDataRec1(2, 5) = 0
        arrDataRec1(2, 4) = 0
        arrDataRec1(2, 3) = 0
        arrDataRec1(2, 2) = 0
        arrDataRec1(2, 2) = 0
        chtRec(1).ChartData = arrDataRec1
    End With
  End With
  '   muestra al comienzo la pestaña 0
  SSTabRes(2).Tab = 0
End Sub


Private Sub cmdApply_Click()
  '   Abre una instancia del Block de Notas con el fichero de salida de la simulación
  Shell "notepad " & SimActual.Result.Archivo, vbNormalFocus
End Sub

Private Sub cmdCancel_Click()
  Unload Me
End Sub

Private Sub cmdOK_Click()
  '   MsgBox "Coloque código aquí para establecer opciones y cerrar el cuadro de diálogo"
  Unload Me
End Sub

Private Sub Form_Deactivate()
  Unload Me
End Sub

Private Sub Form_Load()
  '   centrar el formulario
  Me.Move (Screen.Width - Me.Width) / 2, (Screen.Height - Me.Height) / 2
End Sub

Private Sub Form_LostFocus()
  '   Si el formulario pierde el foco, se descarga
  Unload Me
End Sub


Private Sub Frame4_Click(Index As Integer)
  '   Al hacer click sobre este frame cambia el tipo de representación del gráfico
  '   que contiene entre barras apiladas y sectores
  If Modo Then
    Modo = False
    chtEnv(Index).chartType = VtChChartType2dBar
    chtEnv(Index).Stacking = True
  Else
    Modo = True
    chtEnv(Index).chartType = VtChChartType2dPie
  End If
End Sub

Private Sub Frame5_Click(Index As Integer)
  '   Al hacer click sobre este frame cambia el tipo de representación del gráfico
  '   que contiene entre barras apiladas y sectores
  If Modo Then
    Modo = False
    chtRec(Index).chartType = VtChChartType2dBar
    chtRec(Index).Stacking = True
  Else
    Modo = True
    chtRec(Index).chartType = VtChChartType2dPie
  End If
End Sub
```

6 Formulario Resultados de la batería *frmResSim*.



```
Option Explicit

'   Estos valores pon Public para que puedan ser utilizados como parámetros
'   desde el exterior de este formulario
'   Parámetro representado en el eje X
Public EjeX As String
'   Parámetro representado en el eje X
Public EjeY As String

Private Sub Form_Activate()
    '   Inicializa los valores del formulario

    '   variables índice
    Dim i As Integer
    Dim j As Integer
    '   almacenará el número datos a mostrar
    Dim Tot As Integer

    '   cuenta las simulaciones que hay hechas
    For i = 1 To UBound(Module1.Document)
        If Module1.Document(i).mSim.Realizada = True Then
            j = j + 1
        End If
    Next
    '   almacena ese número en Tot y reinicializa j
    Tot = j
    j = 0

    '   rellena los datos del mschart del nuevo formulario
    With MSChart1
        '   establece el tamaño del gráfico a 2 x Tot
        .ColumnCount = 2
        .RowCount = Tot
        .AutoIncrement = True
        '   comienza con la primera columna, de nombre "Nodo 0"
        .Column = 1
        .ColumnLabel = "Nodo 0"
        '   recorre los formularios secundarios y coge los datos
        '   de las simulaciones realizadas
        For i = 1 To UBound(Module1.Document)
            '   si se ha hecho la simulación coge el dato, si no pasa
            If Module1.Document(i).mSim.Realizada = False Then
            Else
                '   establece la posición del dato
                j = j + 1
                .Row = j
                '   escoge el dato en función de que parámetro se está representando
                '   y añade las unidades al rótulo del eje X
                Select Case EjeX
                Case "Probabilidad de error"
                    .RowLabel = Module1.Document(i).mSim.Param.Canal0.Prob
                Case "Semilla"
                    .RowLabel = Module1.Document(i).mSim.Param.General.Semilla
                Case "Ventana de recepción"
                    .RowLabel = Module1.Document(i).mSim.Param.Nodo0.Vrx
                Case "Ventana de transmision"
                    .RowLabel = Module1.Document(i).mSim.Param.Nodo0.Vtrx
                Case "Temporizador"
                    .RowLabel = Module1.Document(i).mSim.Param.Nodo0.TempPpal
                Case "Tamaño máx. de trama"
                    .RowLabel = Module1.Document(i).mSim.Param.Nodo0.Mtu
                Case "Longuitud del enlace"
                    .RowLabel = Module1.Document(i).mSim.Param.Canal0.Lon
                Case "Régimen binario"
                    .RowLabel = Module1.Document(i).mSim.Param.Canal0.Rb
                End Select
            End If
        Next

        '   ahora hace lo mismo con el nodo 1
        '   pero no necesita volver a leer el eje X
        .Column = 2
        .ColumnLabel = "Nodo 1"
    End With

    '   Establece los rótulos de los ejes del gráfico
    Select Case EjeX
        Case "Probabilidad de error"
            EjeX = EjeX & " [%]"
        Case "Temporizador"
            EjeX = EjeX & " [ms]"
        Case "Tamaño máx. de trama"
            EjeX = EjeX & " [oct]"
        Case "Longuitud del enlace"
            EjeX = EjeX & " [m]"
        Case "Régimen binario"
            EjeX = EjeX & " [b/s]"
    End Select
    EjeY = "Rendimiento [%]"

    '   rellena los datos del gráfico con el rendimiento
    List1 = "Rendimiento"

End Sub

Private Sub List1_Click()
    '   Selecciona el parámetro que se representa en la gráfica y calcula la media

    '   variables indice
    Dim i As Integer
    Dim j As Integer
    Dim k As Integer

    '   variables auxiliares
    Dim Tot As Integer
    Dim mSuma(2) As Double
    Dim mAux(2) As Double

    '   inicialización de variables
    mSuma(1) = 0
    mSuma(2) = 0
    mAux(1) = 0
    mAux(2) = 0
    j = 0
```

```
    ' cuenta el número de formularios de simulación abiertos
    For i = 1 To UBound(Module1.Document)
        ' comprueba que se haya realizado la simulación de ese formulario
        If Module1.Document(i).mSim.Realizada = True Then
            j = j + 1
        End If
    Next
    Tot = j

    ' establece los nuevos valores y calcula las medias
    ' segun el valor elegido de la lista desplegable
    With MSChart1
        ' establece el tamaño de la gráfica
        .ColumnCount = 2
        .RowCount = Tot
        ' cada vez que se introduce un dato se pasa al siguiente
        .AutoIncrement = True
        ' comenzamos con la primera columna, correspondiente al nodo 0
        j = 0
        For i = 1 To UBound(Module1.Document)
            ' si no se ha cerrado el formulario, se tiene en cuenta
            If Module1.FState(i).Deleted = False Then
                j = j + 1
                .Row = j
                ' establece el valor correspondiente
                Select Case List1
                Case "Rendimiento"
                    .Column = 1
                    mAux(1) = Module1.Document(i).mSim.Result.ResCan1.Rend / 100
                    .Data = mAux(1)
                    .Column = 2
                    mAux(2) = Module1.Document(i).mSim.Result.ResCan0.Rend / 100
                    .Data = mAux(2)
                Case "Cadencia eficaz"
                    .Column = 1
                    mAux(1) = Module1.Document(i).mSim.Result.ResCan1.Cef / 100
                    .Data = mAux(1)
                    .Column = 2
                    mAux(2) = Module1.Document(i).mSim.Result.ResCan0.Cef / 100
                    .Data = mAux(2)
                Case "Uso de la línea"
                    .Column = 1
                    mAux(1) = Module1.Document(i).mSim.Result.ResCan0.Uso / 100
                    .Data = mAux(1)
                    .Column = 2
                    mAux(2) = Module1.Document(i).mSim.Result.ResCan1.Uso / 100
                    .Data = mAux(2)
                Case "Retardo medio"
                    .Column = 1
                    mAux(1) = Module1.Document(i).mSim.Result.ResCan1.MeanTrip / 100
                    .Data = mAux(1)
                    .Column = 2
                    mAux(2) = Module1.Document(i).mSim.Result.ResCan0.MeanTrip / 100
                    .Data = mAux(2)
                Case "Retardo medio de ida y vuelta"
                    .Column = 1
                    mAux(1) = Module1.Document(i).mSim.Result.ResCan0.MRoundTrip / 100
                    .Data = mAux(1)
                    .Column = 2
                    mAux(2) = Module1.Document(i).mSim.Result.ResCan1.MRoundTrip / 100
                    .Data = mAux(2)
                End Select
                ' va sumando los valores
                mSuma(1) = mSuma(1) + mAux(1)
                mSuma(2) = mSuma(2) + mAux(2)
            End If
        Next
        ' calcula la media como suma total/número de elementos
        mSuma(1) = mSuma(1) / Tot
        mSuma(2) = mSuma(2) / Tot
    End With

    ' muestra las medias redondeando los resultados a dos decimales
    Select Case List1
        Case "Rendimiento"
            Lbl4 = Round(mSuma(1), 2) & " %"
            Lbl5 = Round(mSuma(2), 2) & " %"
            EjeY = "Rendimiento [%]"
        Case "Cadencia eficaz"
            Lbl4 = Round(mSuma(1), 2) & " [b/s]"
            Lbl5 = Round(mSuma(2), 2) & " [b/s]"
            EjeY = "Cadencia eficaz [b/s]"
        Case "Uso de la línea"
            Lbl4 = Round(mSuma(1), 2) & " %"
            Lbl5 = Round(mSuma(2), 2) & " %"
            EjeY = "Uso del enlace [%]"
        Case "Retardo medio"
            Lbl4 = Round(mSuma(1), 2) & " [ms]"
            Lbl5 = Round(mSuma(2), 2) & " [ms]"
            EjeY = "Retardo medio [ms]"
        Case "Retardo medio de ida y vuelta"
            Lbl4 = Round(mSuma(1), 2) & " [ms]"
            Lbl5 = Round(mSuma(2), 2) & " [ms]"
            EjeY = "RMD [ms]"
    End Select

    ' establece el título del gráfico según los parámetros mostrados
    fmeGraf.Caption = EjeY & " frente a " & EjeX

End Sub
```

7 Módulo *Module1*.

```
Option Explicit

' Tipo definido por el usuario para almacenar información sobre los formularios secundarios
Type FormState
    Deleted As Integer  ' indica si se ha cerrado el formulario
    Dirty As Integer    ' para un uso futuro
    Color As Long       ' Para un uso futuro
```

```
End Type

Public fMainForm As frmMain
Public FState()  As FormState          '   Matriz de tipos definidos por el usuario
Public Document() As New frmDocument    '   Matriz de objetos formulario secundario
Public Sims() As New Simulacion         '   Matriz de objetos Simulacion

Function CreaCopia(ActualFrm As frmDocument)
    '   crea una copia exacta (incluidos valores) del formulario que
    '   se pasa pcomo parámetro
   Dim fIndex As Integer

    '   Busca el siguiente índice disponible y muestra el formulario secundario.
   fIndex = FindFreeIndex()
    '   le asigna este indice para poder loclizarlo después
   Document(fIndex).Tag = fIndex
    '   pone un titulo de formulario estandar
   Document(fIndex).Caption = "Document: " & fIndex
    '   copia el objeto Simulacion perteneciente al formulario que se ha pasado
    '   como parámetro
   Document(fIndex).mSim.Param.Copia ActualFrm.mSim
    '   lee los datos del objeto Simulación copiado y los presenta
    '   en el formulario
   Document(fIndex).chkSimNod = 0
   Document(fIndex).chkSimCan = 0
   Document(fIndex).PresentaParam
    '   muestra el nuevo formulario
   Document(fIndex).Show
    '   actualiza los atributos del formulario en FState
   FState(fIndex).Deleted = False
End Function

Public Sub LoadNewDoc()
    '   crea un nuevo formulario para una simulación nueva
   Dim fIndex As Integer

    '   Busca el siguiente índice disponible y muestra el formulario secundario.
   fIndex = FindFreeIndex()
    '   le asigna este indice para poder loclizarlo después
   Document(fIndex).Tag = fIndex
    '   pone un titulo de formulario estandar
   Document(fIndex).Caption = "Document: " & fIndex
    '   lee los datos del objeto Simulación nuevo (vacío) y los presenta
    '   en el formulario
   Document(fIndex).PresentaParam
    '   muestra el nuevo formulario
   Document(fIndex).Show
End Sub

Public Sub OpenDoc(File As String)
    '   crea un nuevo formulario para una simulación almacenada en
    '   el fichero File
   Dim fIndex As Integer

    '   Busca el siguiente índice disponible y muestra el formulario secundario.
   fIndex = FindFreeIndex()
    '   le asigna este indice para poder loclizarlo después
   Document(fIndex).Tag = fIndex
    '   lee los parámetros de la simulación desde el archivo File
   Document(fIndex).mSim.Param.Archivo = File
   Document(fIndex).mSim.Param.Leer
    '   cambia el título del formulario
   Document(fIndex).Caption = File
    '   presenta los parámetros de la simulación en el formulario
   Document(fIndex).PresentaParam
    '   actualiza los atributos del formulario en FState
   FState(fIndex).Dirty = False
    '   muestra el nuevo formulario
   Document(fIndex).Show
End Sub

Function FindFreeIndex() As Integer
    '   busca un índice líbre en la matriz dinámica de formularios secundarios
    '   este índice será uno intermedio si es que se ha cerrado algún formulario
    '   o 1 + el tamaño actual de la matriz

   Dim i As Integer
   Dim ArrayCount As Integer

   ArrayCount = UBound(Document)   '   Devuelve mayor índice de matriz Document

    '   Recorre la matriz de documentos. Si se ha eliminado alguno de los
    '   documentos, devuelve dicho índice.
   For i = 1 To ArrayCount
      If FState(i).Deleted Then
         '   se ha encontrado un índice libre por un formulario cerrado
         FindFreeIndex = i
         FState(i).Deleted = False
         Exit Function
      End If
   Next

    '   Si no se ha eliminado ninguno de los elementos de la matriz de
    '   documentos, incrementa en uno el rango de la matriz de documentos
    '   y de estados, y devuelve el índice del nuevo elemento.
   ReDim Preserve Document(ArrayCount + 1)
   ReDim Preserve FState(ArrayCount + 1)
    '   no había ningún índice libre
   FindFreeIndex = UBound(Document)
End Function

Public Function LeerNumero(NumFich As Integer) As String
    '   Lee un número del fichero apuntado por NumFich y lo devuelve en una cadena
   Dim MiCaracter As String, MiPalabra As String, MiAux As String

    '   en MiPalabra se va formando el número que se lee
    '   y en MiCaracter se recoge el último caracter leído
   Do
      MiPalabra = ""
      MiCaracter = ""
      Do
         '   se ignora el punto ('  .'  )
         If MiCaracter <> "." Then MiPalabra = MiPalabra & MiCaracter
         '   lee el siguiente caracter
```

```
        MiCaracter = Input(1, #NumFich)
      '   se añaden caracteres hasta encontrar un espacio ('   ' ), un tabulador,
      '   un fin de línea o un avance de línea
      Loop While MiCaracter <> " " And MiCaracter <> vbCr _
        And MiCaracter <> vbLf And MiCaracter <> vbTab
      '   si la parabra leída es "#" se desprecia todo lo que queda en esa línea
      '   y se continúa en la siguente línea
      If MiPalabra = "#" Then
        Line Input #NumFich, MiAux
      End If
    '   se busca la siguiente palabra si lo leído es nada o "#"
    Loop While MiPalabra = "#" Or MiPalabra = ""
    '   si se lee "NaN", se sustituye por 0
    If MiPalabra = "NaN" Then MiPalabra = "0"
    '   se establece el valor a devolver como MiPalabra
    LeerNumero = MiPalabra
End Function

Public Function LeerPalabra(NumFich As Integer) As String
    '   Lee la siguiente palabra del fichero apuntado por NumFich
    '   y lo devuelve en una cadena
    Dim MiCaracter As String, MiPalabra As String, MiAux As String

    '   en MiPalabra se va formando el número que se lee
    '   y en MiCaracter se recoge el último caracter leído
    Do
      MiPalabra = ""
      MiCaracter = ""
      Do
        '   va añadiendo caracteres a la palabra
        MiPalabra = MiPalabra & MiCaracter
        '   lee el siguiente caracter
        MiCaracter = Input(1, #NumFich)
      '   se añaden caracteres hasta encontrar un espacio ('   ' ), un tabulador,
      '   un fin de línea o un avance de línea
      Loop While MiCaracter <> " " And MiCaracter <> vbCr _
        And MiCaracter <> vbLf And MiCaracter <> vbTab
      '   si la parabra leída es "#" se desprecia todo lo que queda en esa línea
      '   y se continúa en la siguente línea
      If MiPalabra = "#" Then
        Line Input #NumFich, MiAux
      End If
    '   se busca la siguiente palabra si lo leído es nada o "#"
    Loop While MiPalabra = "#" Or MiPalabra = ""
    '   se establece el valor a devolver como MiPalabra
    LeerPalabra = MiPalabra
End Function

Sub Main()
    '   en esta función se puede introducir un formulario al comienzo del
    '   programa donde se de le bienvenida al usuario

    '   frmSplash.Show
    '   frmSplash.Refresh
    Set fMainForm = New frmMain
    Load fMainForm
    '   Unload frmSplash
    '   muestra el formulario principal
    fMainForm.Show
End Sub
```

8 Módulo de clase *Simulacion*.

```
Option Explicit

'   Este es el Objeto Simulación
'   Se compone de otros dos objetos:
'     - un objeto Param donde se almacenan los parámetros de la simulación
'     - un objeto Result donde se almacenan los resuldatos de la simulación
'   y de dos propiedades:
'     - ID: en un índice
'     - Realizada: un booleano que indica si la simulación se ha realizado
Private mvarParam As Param
Private mvarResult As Result
'  variables locales para almacenar los valores de las propiedades
Private mvarID As String '  copia local
'  variables locales para almacenar los valores de las propiedades
Private mvarRealizada As Boolean '  copia local

Public Property Let Realizada(ByVal vData As Boolean)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Realizada = 5
    mvarRealizada = vData
End Property


Public Property Get Realizada() As Boolean
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Realizada
    Realizada = mvarRealizada
End Property



Public Property Let ID(ByVal vData As String)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.ID = 5
    mvarID = vData
End Property


Public Property Get ID() As String
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.ID
    ID = mvarID
End Property



Public Property Get Result() As Result
```

```
    Set Result = mvarResult
End Property


Public Property Set Result(vData As Result)
    Set mvarResult = vData
End Property

Public Property Get Param() As Param
    Set Param = mvarParam
End Property
Public Property Set Param(vData As Param)
    Set mvarParam = vData
End Property
Public Sub Class_Initialize()
  ' crear el objeto mParam cuando se crea la clase Simulacion
  Set mvarParam = New Param
  ' crear el objeto mResult cuando se crea la clase Simulacion
  Set mvarResult = New Result
End Sub
Public Sub Class_Terminate()
  Set mvarResult = Nothing
  Set mvarParam = Nothing
End Sub
```

9 Módulo de clase *Param*.

```
Option Explicit

'   Este es el objeto Param donde se almacenan los parámetros de la simulación
'   Se compone de otros cinco objetos:
'     - General: es un objeto General que almacena los parámetros generales
'                de la simulación.
'     - Nodo0: es un objeto Nodo que almacena los parámetros del nodo 0
'     - Nodo1: es un objeto Nodo que almacena los parámetros del nodo 1
'     - Canal0: es un objeto Nodo que almacena los parámetros del canal 0
'     - Canal1: es un objeto Nodo que almacena los parámetros del canal 1
'   de una propiedad:
'     - Archivo: una string que almacena el nombre del fichero de simulación
'     - y de
'   y de tres métodos:
'     - Copia(): copia los datos del objeto Param que se le pasa como parámetro
'     - Escribir(): crea un archivo de simulación con los datos del objeto
'     - Leer(): lee los datos de un archivo de simulación

'   variables locales para almacenar los valores de las propiedades
Private mvarGeneral As General
Private mvarNodo0 As Nodo
Private mvarNodo1 As Nodo
Private mvarCanal0 As Canal
Private mvarCanal1 As Canal
Private mvarArchivo As String

Public Property Let Archivo(ByVal vData As String)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Archivo = 5
    mvarArchivo = vData
End Property
Public Property Get Archivo() As String
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Archivo
    Archivo = mvarArchivo
End Property
Public Property Get Canal0() As Canal
    Set Canal0 = mvarCanal0
End Property
Public Property Get Canal1() As Canal
    Set Canal1 = mvarCanal1
End Property
Public Property Set Canal1(vData As Canal)
    Set mvarCanal1 = vData
End Property
Public Property Set Canal0(vData As Canal)
    Set mvarCanal0 = vData
End Property
Public Sub Copia(OtraSim As Simulacion)
    '   Copia los datos correspondientes al objeto Param perteneciente al
    '   objeto OtraSim que le pasan como parámetro

    With OtraSim.Param
        '   los datos del canal 0
        With .Canal0
            Canal0.Rb = .Rb
            Canal0.Vp = .Vp
            Canal0.Lon = .Lon
            Canal0.Prob = .Prob
        End With
        '   los datos del canal 1
        With .Canal1
            Canal1.Rb = .Rb
            Canal1.Vp = .Vp
            Canal1.Lon = .Lon
            Canal1.Prob = .Prob
        End With
        '   los datos del nodo 0
        With .Nodo0
            Nodo0.TempPpal = .TempPpal
            Nodo0.TempAck = .TempAck
            Nodo0.Traza = .Traza
            Nodo0.Vtrx = .Vtrx
            Nodo0.Vrx = .Vrx
            Nodo0.Mtu = .Mtu
            Nodo0.FichLeer = .FichLeer
            Nodo0.FichEsc = .FichEsc
        End With
        '   los datos del nodo 1
        With .Nodo1
            Nodo1.TempPpal = .TempPpal
            Nodo1.TempAck = .TempAck
            Nodo1.Traza = .Traza
            Nodo1.Vtrx = .Vtrx
            Nodo1.Vrx = .Vrx
```

```
        Nodo1.Mtu = .Mtu
        Nodo1.FichLeer = .FichLeer
        Nodo1.FichEsc = .FichEsc
      End With
      '   los datos generales
      With .General
        General.Ttotal = .Ttotal
        General.TickSeg = .TickSeg
        General.FichSal = .FichSal
        General.Semilla = .Semilla
        General.Acks = .Acks
        General.Nacks = .Nacks
      End With
    End With
End Sub


Public Sub Escribir()
    '   Escribe los datos contenidos en el objeto en el archivo cuyo nombre
    '   es la propiedad Archivo

    Dim fso, txtfile, aux

    '   crea el archivo Archivo
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set txtfile = fso.CreateTextFile(Archivo, True)

    '   comienza a escribir en el archivo apuntado por txtfile
    '   ha de respetar la estructura que marca el simulador sim.exe
    With txtfile
        '   Escribe una línea de comentario
      .WriteLine (General.Descr)
      '   una línea en blanco
      .WriteBlankLines (1)
      '   la palabra "PARAMETROS" en una línea
      .WriteLine ("PARAMETROS")
      '   una línea en blanco
      .WriteBlankLines (1)
      '   la palabra "SIMULACION" en una línea
      .WriteLine ("SIMULACION")
      '   y a continuación cada uno de los parámetros en una línea cada uno
      '   con un par de tabuladores entre el nombre y el valor
      .WriteLine ("tiempo_total" & vbTab & vbTab & General.Ttotal)
      .WriteLine ("ticks_seg" & vbTab & vbTab & General.TickSeg)
      .WriteLine ("fichero_salida" & vbTab & vbTab & General.FichSal)
      .WriteLine ("semilla" & vbTab & vbTab & vbTab & General.Semilla)
      '   acks es distinto porque la propiedad es un booleano pero el simulador
      '   necesita un '  0'    o un '  1'
      .Write ("acks" & vbTab & vbTab & vbTab)
      If General.Acks Then
        .WriteLine ("1")
      Else
        .WriteLine ("0")
      End If
      '   lo mismo le sucede a nacks
      .Write ("nacks" & vbTab & vbTab & vbTab)
      If General.Nacks Then
        .WriteLine ("1")
      Else
        .WriteLine ("0")
      End If
      '   fin de los parámetros generales
      .WriteLine ("/SIMULACION")
      .WriteBlankLines (1)
      '   comienzan los parámetros de canal 0
      .WriteLine ("CANAL 0")
      .WriteLine ("regimen_binario" & vbTab & vbTab & Canal0.Rb)
      .WriteLine ("velocidad_propagacion" & vbTab & Canal0.Vp)
      .WriteLine ("longitud" & vbTab & vbTab & Canal0.Lon)
      .WriteLine ("error_trama" & vbTab & vbTab & Canal0.Prob)
      .WriteLine ("/CANAL")
      .WriteBlankLines (1)
      '   ahora los parámetros de canal 1
      .WriteLine ("CANAL 1")
      .WriteLine ("regimen_binario" & vbTab & vbTab & Canal1.Rb)
      .WriteLine ("velocidad_propagacion" & vbTab & Canal1.Vp)
      .WriteLine ("longitud" & vbTab & vbTab & Canal1.Lon)
      .WriteLine ("error_trama" & vbTab & vbTab & Canal1.Prob)
      .WriteLine ("/CANAL")
      .WriteBlankLines (1)
      '   ahora los parámetros de nodo 0
      .WriteLine ("NODO 0")
      .WriteLine ("temporizador" & vbTab & vbTab & Nodo0.TempPpal)
      .WriteLine ("temporizador_ack" & vbTab & Nodo0.TempAck)
      .WriteLine ("traza" & vbTab & vbTab & vbTab & Nodo0.Traza)
      .WriteLine ("ventana_trx" & vbTab & vbTab & Nodo0.Vtrx)
      .WriteLine ("ventana_rx" & vbTab & vbTab & Nodo0.Vrx)
      .WriteLine ("mtu" & vbTab & vbTab & vbTab & Nodo0.Mtu)
      '   si el dato FichLeer está vacío no hace falta escribirlo en el archivo
      If Nodo0.FichLeer <> "" Then
        .WriteLine ("fichero_enviar" & vbTab & vbTab & Nodo0.FichLeer)
      End If
      '   lo mismo le pasa al dato FichEsc
      If Nodo0.FichEsc <> "" Then
        .WriteLine ("fichero_recibir" & vbTab & vbTab & Nodo0.FichEsc)
      End If
      .WriteLine ("/NODO")
      .WriteBlankLines (1)
      '   ahora los parámetros de nodo 1
      .WriteLine ("NODO 1")
      .WriteLine ("temporizador" & vbTab & vbTab & Nodo1.TempPpal)
      .WriteLine ("temporizador_ack" & vbTab & Nodo1.TempAck)
      .WriteLine ("traza" & vbTab & vbTab & vbTab & Nodo1.Traza)
      .WriteLine ("ventana_trx" & vbTab & vbTab & Nodo1.Vtrx)
      .WriteLine ("ventana_rx" & vbTab & vbTab & Nodo1.Vrx)
      .WriteLine ("mtu" & vbTab & vbTab & vbTab & Nodo1.Mtu)
      If Nodo1.FichLeer <> "" Then
        .WriteLine ("fichero_enviar" & vbTab & vbTab & Nodo1.FichLeer)
      End If
      If Nodo1.FichEsc <> "" Then
        .WriteLine ("fichero_recibir" & vbTab & vbTab & Nodo1.FichEsc)
      End If
      .WriteLine ("/NODO")
      .WriteBlankLines (1)
```

```
      '   fin del fichero
      .WriteLine ("/PARAMETROS")
      .Close
   End With
End Sub
Public Property Get Nodo1() As Nodo
   Set Nodo1 = mvarNodo1
End Property
Public Property Get Nodo0() As Nodo
   Set Nodo0 = mvarNodo0
End Property
Public Property Set Nodo1(vData As Nodo)
   Set mvarNodo1 = vData
End Property
Public Property Set Nodo0(vData As Nodo)
   Set mvarNodo0 = vData
End Property
Public Property Get General() As General
   Set General = mvarGeneral
End Property
Public Property Set General(vData As General)
   Set mvarGeneral = vData
End Property
Public Sub Leer()
   '   Lee del archivo de simulación Archivo los datos de la simulación

   Dim MiNumero As Integer
   Dim MiCadena As String, Descr As String

   '   Abre el archivo para recibir los datos.
   Open Archivo For Input As #1
   '   si hay algún error se muestra el mensaje correspondiente
   If Err Then
      MsgBox "No se puede abrir el archivo: " + Archivo
      Exit Sub
   End If

   '   lee la primera línea y la guarda en el dato Descr
   Line Input #1, MiCadena
   General.Descr = MiCadena
   '   lee la sigiente palabra
   MiCadena = LeerPalabra(1)
   '   si la palabra leída es "PARAMETROS" es que vamos bien
   If MiCadena = "PARAMETROS" Then
      '   lee la sigiente palabra
      MiCadena = LeerPalabra(1)
      '   según la palabra leida sabemos que tipo de datos vamos a leer
      Do
         '   datos generales
         If MiCadena = "SIMULACION" Then
            Call LeerSimulacion(1)
         '   datos de nodo
         ElseIf MiCadena = "NODO" Then
            Call LeerNodo(1)
         '   datos de canal
         ElseIf MiCadena = "CANAL" Then
            Call LeerCanal(1)
         End If
         MiCadena = LeerPalabra(1)
      '   sigue leyendo hasta encontrar el fin de fichero
      Loop While MiCadena <> "/PARAMETROS"
   End If
   '   Cierra el archivo.
   Close #1
End Sub
Private Sub LeerSimulacion(File As Integer)
   '   lee los parametros generales de la simulación y los guarda en el
   '   objeto General

   Dim MiPalabra As String

   '   según la palabra leída sabemos que dato vamos a leer
   MiPalabra = LeerPalabra(File)
   Do
      '   tiempo total de simulación
      If MiPalabra = "tiempo_total" Then
         General.Ttotal = LeerPalabra(File)
      '   número de ticks por segundo
      ElseIf MiPalabra = "ticks_seg" Then
         General.TickSeg = LeerPalabra(File)
      '   fichero de salida
      ElseIf MiPalabra = "fichero_salida" Then
         General.FichSal = LeerPalabra(File)
      '   semilla
      ElseIf MiPalabra = "semilla" Then
         General.Semilla = LeerPalabra(File)
      '   acks, hay que tener cuidado al leer este dato
      ElseIf MiPalabra = "acks" Then
         If LeerPalabra(File) Then
            General.Acks = True
         Else
            General.Acks = False
         End If
      '   nacks, como el anterior
      ElseIf MiPalabra = "nacks" Then
         If LeerPalabra(File) Then
            General.Nacks = True
         Else
            General.Nacks = False
         End If
      End If
      MiPalabra = LeerPalabra(File)
   '   sigue leyendo hasa encontrar la marca de fin de parámetros generales
   Loop While MiPalabra <> "/SIMULACION"
End Sub
Private Sub LeerNodo(File As Integer)
   '   lee los parametros de nodo de la simulación y los guarda en el
   '   objeto Nodo correspondiente

   Dim MiPalabra As String
   Dim Index As Integer
   Dim Nod As Nodo

   '   lee la siguiente palabra que es el índice de nodo
```

```
    Index = LeerPalabra(File)
    '   según en índice, se rellenara el objeto Nodo0 o Nodo1
    If Index = 0 Then Set Nod = Nodo0 Else Set Nod = Nodo1
    MiPalabra = LeerPalabra(File)
    '   según la palabra leída sabemos que dato vamos a leer
    With Nod
    Do
        '   temporizador principal
        If MiPalabra = "temporizador" Then
            .TempPpal = LeerPalabra(File)
        '   temporizador de asentimiento
        ElseIf MiPalabra = "temporizador_ack" Then
            .TempAck = LeerPalabra(File)
        '   traza
        ElseIf MiPalabra = "traza" Then
            .Traza = LeerPalabra(File)
        '   ventana de transmisión
        ElseIf MiPalabra = "ventana_trx" Then
            .Vtrx = LeerPalabra(File)
        '   ventana de recepción
        ElseIf MiPalabra = "ventana_rx" Then
            .Vrx = LeerPalabra(File)
        '   tamaño máximo de trama
        ElseIf MiPalabra = "mtu" Then
            .Mtu = LeerPalabra(File)
        '   fichero a enviar
        ElseIf MiPalabra = "fichero_enviar" Then
            .FichLeer = LeerPalabra(File)
        '   fichero a recibir
        ElseIf MiPalabra = "fichero_recibir" Then
            .FichEsc = LeerPalabra(File)
        End If
        MiPalabra = LeerPalabra(File)
    '   sigue leyendo hasa encontrar la marca de fin de parámetros de nodo
    Loop While MiPalabra <> "/NODO"
    End With
End Sub
Private Sub LeerCanal(File As Integer)
    '   lee los parametros de canal de la simulación y los guarda en el
    '   objeto Canal correspondiente

    Dim MiPalabra As String
    Dim Index As Integer
    Dim Can As Canal

    '   lee la siguiente palabra que es el índice de canal
    Index = LeerPalabra(File)
    '   según en índice, se rellenara el objeto Canal0 o Canal1
    If Index = 0 Then Set Can = Canal0 Else Set Can = Canal1
    MiPalabra = LeerPalabra(File)
    '   según la palabra leída sabemos que dato vamos a leer
    With Can
    Do
        '   regimen binario
        If MiPalabra = "regimen_binario" Then
            .Rb = LeerPalabra(File)
        '   velocidad de propagación
        ElseIf MiPalabra = "velocidad_propagacion" Then
            .Vp = LeerPalabra(File)
        '   longitud del enlace
        ElseIf MiPalabra = "longitud" Then
            .Lon = LeerPalabra(File)
        '   probabilidad de error de trama
        ElseIf MiPalabra = "error_trama" Then
            .Prob = LeerPalabra(File)
        End If
        MiPalabra = LeerPalabra(File)
    '   sigue leyendo hasa encontrar la marca de fin de parámetros de nodo
    Loop While MiPalabra <> "/CANAL"
    End With
End Sub
Private Sub Class_Initialize()
    '   crear el objeto mGeneral cuando se crea la clase Simulacion
    Set mvarGeneral = New General
    '   crear el objeto mCanal0 cuando se crea la clase Simulacion
    Set mvarCanal0 = New Canal
    '   crear el objeto mCanal1 cuando se crea la clase Simulacion
    Set mvarCanal1 = New Canal
    '   crear el objeto mNodo0 cuando se crea la clase Simulacion
    Set mvarNodo0 = New Nodo
    '   crear el objeto mNodo1 cuando se crea la clase Simulacion
    Set mvarNodo1 = New Nodo
End Sub
Private Sub Class_Terminate()
    Set mvarNodo1 = Nothing
    Set mvarNodo0 = Nothing
    Set mvarCanal1 = Nothing
    Set mvarCanal0 = Nothing
    Set mvarGeneral = Nothing

End Sub
```

10 Módulo de clase *General*.

```
Option Explicit

'   Este es el objeto General, que almacena los parámetros generales de la simulación
'   consta de siete propiedades:
'       - Descr: descripción de la simulación
'       - Ttotal: durección total de la simulación
'       - TickSeg: número de ticks por segundo
'       - Semilla: semilla de la simulación
'       - FichSal: fichero donde se almacenará la salida del simulador
'       - Acks: si se van a a utilizar acks o no
'       - Nacks: si se van a a utilizar nacks o no

'  variables locales para almacenar los valores de las propiedades
Private mvarDescr As String '  copia local
Private mvarTtotal As Integer '  copia local
Private mvarTickSeg As Integer '  copia local
Private mvarSemilla As Integer '  copia local
Private mvarFichSal As String '  copia local
```

```
Private mvarAcks As Boolean '  copia local
Private mvarNacks As Boolean '  copia local
Public Property Let Nacks(ByVal vData As Boolean)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Nacks = 5
    mvarNacks = vData
End Property
Public Property Get Nacks() As Boolean
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Nacks
    Nacks = mvarNacks
End Property
Public Property Let Acks(ByVal vData As Boolean)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Acks = 5
    mvarAcks = vData
End Property
Public Property Get Acks() As Boolean
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Acks
    Acks = mvarAcks
End Property
Public Property Let FichSal(ByVal vData As String)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.FichSal = 5
    mvarFichSal = vData
End Property
Public Property Get FichSal() As String
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.FichSal
    FichSal = mvarFichSal
End Property
Public Property Let Semilla(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Semilla = 5
    mvarSemilla = vData
End Property
Public Property Get Semilla() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Semilla
    Semilla = mvarSemilla
End Property
Public Property Let TickSeg(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.TickSeg = 5
    mvarTickSeg = vData
End Property
Public Property Get TickSeg() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.TickSeg
    TickSeg = mvarTickSeg
End Property
Public Property Let Ttotal(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Ttotal = 5
    mvarTtotal = vData
End Property
Public Property Get Ttotal() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Ttotal
    Ttotal = mvarTtotal
End Property
Public Property Let Descr(ByVal vData As String)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Descr = 5
    mvarDescr = vData
End Property
Public Property Get Descr() As String
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Descr
    Descr = mvarDescr
End Property
```

11 Módulo de clase *Nodo*.

```
Option Explicit

'   Este es el objeto Nodo, que almacena los parámetros de un nodo de la simulación
'   consta de ocho propiedades:
'      - TempPpal: temporizador principal
'      - TempAck:  temporizador de asentimiento
'      - Vtrx: ventan de transmisión
'      - Vrx: ventana de recepción
'      - FichLeer: fichero a leer
'      - FichEsc: fichero a escribir
'      - Mtu: tamaño máximo de trama
'      - Traza: traza

'  variables locales para almacenar los valores de las propiedades
Private mvarTempPpal As Integer '  copia local
Private mvarTempAck As Integer '  copia local
Private mvarVtrx As Integer '  copia local
Private mvarVrx As Integer '  copia local
Private mvarTraza As Integer '  copia local
Private mvarFichLeer As String '  copia local
Private mvarFichEsc As String '  copia local
Private mvarMtu As Integer '  copia local
Public Property Let Mtu(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Mtu = 5
    mvarMtu = vData
End Property


Public Property Get Mtu() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Mtu
    Mtu = mvarMtu
End Property
```

```
Public Property Let FichEsc(ByVal vData As String)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.FichEsc = 5
    mvarFichEsc = vData
End Property


Public Property Get FichEsc() As String
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.FichEsc
    FichEsc = mvarFichEsc
End Property



Public Property Let FichLeer(ByVal vData As String)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.FichLeer = 5
    mvarFichLeer = vData
End Property


Public Property Get FichLeer() As String
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.FichLeer
    FichLeer = mvarFichLeer
End Property



Public Property Let Traza(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Traza = 5
    mvarTraza = vData
End Property


Public Property Get Traza() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Traza
    Traza = mvarTraza
End Property



Public Property Let Vrx(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Vrx = 5
    mvarVrx = vData
End Property


Public Property Get Vrx() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Vrx
    Vrx = mvarVrx
End Property



Public Property Let Vtrx(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Vtrx = 5
    mvarVtrx = vData
End Property


Public Property Get Vtrx() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Vtrx
    Vtrx = mvarVtrx
End Property



Public Property Let TempAck(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.TempAck = 5
    mvarTempAck = vData
End Property


Public Property Get TempAck() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.TempAck
    TempAck = mvarTempAck
End Property



Public Property Let TempPpal(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.TempPpal = 5
    mvarTempPpal = vData
End Property


Public Property Get TempPpal() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.TempPpal
    TempPpal = mvarTempPpal
End Property
```

12 Módulo de clase *Canal*.

```
Option Explicit

'   Este es el objeto Canal, que almacena los parámetros de un canal de la simulación
'   consta de cuatro propiedades:
'      - Rb: régimen binario
'      - Vp: velocidad de propagación
'      - Lon: longitud del enlace
```

```
'     - Prob: probabilidad de error de trama

'  variables locales para almacenar los valores de las propiedades
Private mvarRb As Long '  copia local
Private mvarVp As Long '  copia local
Private mvarLon As Long '  copia local
Private mvarProb As Integer '  copia local
Public Property Let Prob(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Prob = 5
    mvarProb = vData
End Property
Public Property Get Prob() As Integer
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Prob
    Prob = mvarProb
End Property
Public Property Let Lon(ByVal vData As Long)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Lon = 5
    mvarLon = vData
End Property
Public Property Get Lon() As Long
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Lon
    Lon = mvarLon
End Property
Public Property Let Vp(ByVal vData As Long)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Vp = 5
    mvarVp = vData
End Property
Public Property Get Vp() As Long
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Vp
    Vp = mvarVp
End Property
Public Property Let Rb(ByVal vData As Long)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Rb = 5
    mvarRb = vData
End Property
Public Property Get Rb() As Long
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Rb
    Rb = mvarRb
End Property
```

13 Módulo de clase *Result*.

```
Option Explicit

'   Este es el objeto Result donde se almacenan los resultados de la simulación
'   Se compone de otros cuatro objetos:
'     - ResNod0: es un objeto ResNod que almacena los resultados del nodo 0
'     - ResCan0: es un objeto ResCan que almacena los resultados del canal 0
'     - ResNod1: es un objeto ResNod que almacena los resultados del nodo 1
'     - ResCan0: es un objeto ResCan que almacena los resultados del canal 1
'   de tres propiedades:
'     - Archivo: una string que almacena el nombre del fichero de salida del simulador
'     - Ttotal: almacena el tiempo total simulado
'     - Tsim: almacena el tiempo que se ha empleado en realizar la simulación
'   y de un método:
'     - Leer(): lee los datos del un archivo de salida del simulador

'  variables locales para almacenar los valores de las propiedades
Private mvarResNod0 As ResNod
Private mvarResCan0 As ResCan
Private mvarResNod1 As ResNod
Private mvarResCan1 As ResCan
Private mvarArchivo As String
Private mvarTtotal As Single
Private mvarTsim As Single
Public Property Let Tsim(ByVal vData As Single)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Tsim = 5
    mvarTsim = vData
End Property
Public Property Get Tsim() As Single
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Tsim
    Tsim = mvarTsim
End Property
Public Property Let Ttotal(ByVal vData As Single)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Ttotal = 5
    mvarTtotal = vData
End Property
Public Property Get Ttotal() As Single
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Ttotal
    Ttotal = mvarTtotal
End Property
Private Sub Busca(Fich As Integer, Cadena As String)
    Dim MiPalabra As String

    Do
        MiPalabra = LeerPalabra(Fich)
    Loop While MiPalabra <> Cadena
End Sub
Public Function Leer() As Boolean
    '   Lee los resultados de la simulación recogidos en el archivo de salida
    '   del simualdor sim.exe llamado Archivo
    '   devuelve un boolean que indica el éxito o fracaso de la lectura

    Dim MiNumero As Integer
    Dim MiCadena As String

    '   Abre el archivo para recibir los datos.
    On Error Resume Next
    Open Archivo For Input As #2
    '   si no hay error, procede
```

```
  If Err = 0 Then
     ' borra el registro de errores
     On Error GoTo 0
     ' busca el comienzo de los resultados
     MiCadena = BuscaResult(2)
     ' según la ultima palabra leída sabemos que resultados vamos a leer
     Do
        ' resultados de nodos y canales
        If MiCadena = "Nodo" Then
           Call LeerProcess(2)
        ' resultados generales
        ElseIf MiCadena = "General" Then
           Call LeerGen(2)
        End If
        MiCadena = LeerPalabra(2)
     ' sigue leyendo hasta encontral la cadena "FIN" o el final de fichero
     Loop While MiCadena <> "Fin" Or EOF(2)
     ' si lo último leído ha sido "FIN" no ha habido ningún error
     If MiCadena = "Fin" Then
        Leer = True
     Else
        ' se ha producido algún error
        Leer = False
     End If
     ' Cierra el archivo.
     Close #2
  Else
     ' se ha producido algún error
     Leer = False
  End If
End Function
Private Function BuscaResult(NumFich As Integer) As String
  ' Busca el comienzo de los resultados, este se marca con una línea en blanco

  Dim MiLinea As String

  ' va leyendo el fichero línea a línea
  Do
     MiLinea = ""
     Line Input #NumFich, MiLinea
  ' deja de leer cuando encuentra una línea vacía
  Loop While Not (MiLinea Like "")
  ' establece el valor a devolver
  BuscaResult = MiLinea
End Function
Private Sub LeerProcess(Fich As Integer)
  ' Lee los resultados de los nodos y los canales, una vez situados estos en el
  ' fichero de salida del simulador

  Dim MiInt As Integer
  Dim MiResCan As ResCan, MiResNod As ResNod
  Dim str As String

  ' lee el índice, para distingir entre los resultados del nodo y canal 0 ó el 1
  MiInt = LeerPalabra(Fich)
  If MiInt = 0 Then
     Set MiResCan = ResCan0
     Set MiResNod = ResNod0
  ElseIf MiInt = 1 Then
     Set MiResCan = ResCan1
     Set MiResNod = ResNod1
  End If

  ' para encontrar los valores va buscando ' :' , y detrás estarán los
  ' valores buscados, siempre en orden
  Call Busca(Fich, ":")
  ' resultados del nodo
  With MiResNod
     Call Busca(Fich, ":")
     ' tramas enviadas
     .DatosEnv = LeerPalabra(Fich)
     Call Busca(Fich, ":")
     ' tramas retransmitidas
     .Retr = LeerPalabra(Fich)
     Call Busca(Fich, ":")
     ' asentimientos enviados
     .AckEnv = LeerPalabra(Fich)
     Call Busca(Fich, ":")
     ' asentimientos recibidos sin error
     .AckOK = LeerPalabra(Fich)
     Call Busca(Fich, ":")
     ' asentimientos recibidos con error
     .AckMal = LeerPalabra(Fich)
     Call Busca(Fich, ":")
     ' tramas de datos recibidas sin error
     .DatOK = LeerPalabra(Fich)
     Call Busca(Fich, ":")
     ' tramas de datos recibidas con error
     .DatMal = LeerPalabra(Fich)
     Call Busca(Fich, ":")
     ' paquetes entregados al nivel 3
     .Pkt = LeerPalabra(Fich)
     Call Busca(Fich, ":")
     ' número de veces que ha saltado el temporizador principal
     .Tim = LeerPalabra(Fich)
     Call Busca(Fich, ":")
     ' número de veces que ha saltado el temporizador de asentimiento
     .TimAck = LeerPalabra(Fich)
     ' realiza los calculos necesario para rellenar el resto de los datos
     ' número total de tramas recibidas
     .TotRec = .AckOK + .AckMal + .DatMal + .DatOK
     ' número total de tramas enviadas
     .TotEnv = .DatosEnv + .AckEnv
     ' tramas de datos enviadas sin contar retransmisiones
     .DatosEnv = .DatosEnv - .Retr
  End With
  ' resultados del canal
  With MiResCan
     Call Busca(Fich, ":")
     ' retardo medio
     .MeanTrip = LeerNumero(Fich)
     Call Busca(Fich, ":")
     ' retardo medio de vualta
     .MRoundTrip = LeerNumero(Fich)
```

```
      Call Busca(Fich, ":")
      '   uso del enlace
      .Uso = LeerNumero(Fich)
      Call Busca(Fich, ":")
      '   rendimiento
      .Rend = LeerNumero(Fich)
      Call Busca(Fich, ":")
      '   cadencia eficaz
      .Cef = LeerNumero(Fich)
   End With
End Sub
Private Sub LeerGen(Fich As Integer)
   '   Lee los resultados generales de la simulación, una vez situados estos en el
   '   fichero de salida del simulador

   '   aún hay que buscar ' :'   para llegar a los datos generales (ver estructura
   '   del fichero de salida). Los resultados generales están precedidos por un '  ='
   '   en vez de un '  :'
   Call Busca(Fich, ":")
   Call Busca(Fich, "=")
   '   tiempo que ha durado la simulacion
   Tsim = LeerNumero(Fich) / 100
   Call Busca(Fich, "=")
   '   tiempo total simulado
   Ttotal = LeerNumero(Fich) / 100
End Sub
Public Property Get ResCan0() As ResCan
   Set ResCan0 = mvarResCan0
End Property
Public Property Set ResCan0(vData As ResCan)
   Set mvarResCan0 = vData
End Property
Public Property Get ResCan1() As ResCan
   Set ResCan1 = mvarResCan1
End Property
Public Property Set ResCan1(vData As ResCan)
   Set mvarResCan1 = vData
End Property
Public Property Let Archivo(ByVal vData As String)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Archivo = 5
   mvarArchivo = vData
End Property
Public Property Get Archivo() As String
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Archivo
   Archivo = mvarArchivo
End Property
Private Sub Class_Initialize()
   '  crear el objeto mResNod cuando se crea la clase Result
   Set mvarResNod1 = New ResNod
   Set mvarResNod0 = New ResNod
   Set mvarResCan1 = New ResCan
   Set mvarResCan0 = New ResCan
End Sub
Public Property Get ResNod1() As ResNod
   Set ResNod1 = mvarResNod1
End Property
Public Property Set ResNod1(vData As ResNod)
   Set mvarResNod1 = vData
End Property
Public Property Get ResNod0() As ResNod
   Set ResNod0 = mvarResNod0
End Property
Public Property Set ResNod0(vData As ResNod)
   Set mvarResNod0 = vData
End Property
Private Sub Class_Terminate()
   Set mvarResCan0 = Nothing
   Set mvarResCan1 = Nothing
   Set mvarResNod0 = Nothing
   Set mvarResNod1 = Nothing
End Sub
```

14 Módulo de clase *ResNod*.

```
Option Explicit

'   Este es el objeto ResNod, que almacena los resultados de un nodo de la simulación
'   consta de doce propiedades:
'      - TotEnv: total de tramas enviadas
'      - DatosEnv: total de datos enviados, sin retransmisiones
'      - AckEnv: asentimiento enviados
'      - Tim: número de veces que ha saltado el temporizador principal
'      - Retr: retransmisiones realizadas
'      - TotRec: total de tramas recibidas
'      - DatOK: tramas de datos recibidas sin error
'      - DatMal: tramas de datos recibidas con error
'      - AckOK: tramas de asentimiento recibidas sin error
'      - AckMal: tramas de asentimiento recibidas con error
'      - Pkt: paquetes entregados al nivel 3
'      - TimAck:  número de veces que ha saltado el temporizador de asentimientos

'  variables locales para almacenar los valores de las propiedades
Private mvarTotEnv As Integer '  copia local
Private mvarDatosEnv As Integer '  copia local
Private mvarAckEnv As Integer '  copia local
Private mvarTim As Integer '  copia local
Private mvarRetr As Integer '  copia local
Private mvarTotRec As Integer '  copia local
Private mvarDatOK As Integer '  copia local
Private mvarDatMal As Integer '  copia local
Private mvarAckOK As Integer '  copia local
Private mvarAckMal As Integer '  copia local
Private mvarPkt As Integer '  copia local
Private mvarTimAck As Integer '  copia local
Public Property Let TimAck(ByVal vData As Integer)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.TimAck = 5
   mvarTimAck = vData
End Property
Public Property Get TimAck() As Integer
```

```
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.TimAck
   TimAck = mvarTimAck
End Property
Public Property Let Pkt(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.Pkt = 5
   mvarPkt = vData
End Property
Public Property Get Pkt() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.Pkt
   Pkt = mvarPkt
End Property
Public Property Let AckMal(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.AckMal = 5
   mvarAckMal = vData
End Property
Public Property Get AckMal() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.AckMal
   AckMal = mvarAckMal
End Property
Public Property Let AckOK(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.AckOK = 5
   mvarAckOK = vData
End Property
Public Property Get AckOK() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.AckOK
   AckOK = mvarAckOK
End Property
Public Property Let DatMal(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.DatMal = 5
   mvarDatMal = vData
End Property
Public Property Get DatMal() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.DatMal
   DatMal = mvarDatMal
End Property
Public Property Let DatOK(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.DatOK = 5
   mvarDatOK = vData
End Property
Public Property Get DatOK() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.DatOK
   DatOK = mvarDatOK
End Property
Public Property Let TotRec(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.TotRec = 5
   mvarTotRec = vData
End Property
Public Property Get TotRec() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.TotRec
   TotRec = mvarTotRec
End Property
Public Property Let Retr(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.Retr = 5
   mvarRetr = vData
End Property
Public Property Get Retr() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.Retr
   Retr = mvarRetr
End Property
Public Property Let Tim(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.Tim = 5
   mvarTim = vData
End Property
Public Property Get Tim() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.Tim
   Tim = mvarTim
End Property
Public Property Let AckEnv(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.AckEnv = 5
   mvarAckEnv = vData
End Property
Public Property Get AckEnv() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.AckEnv
   AckEnv = mvarAckEnv
End Property
Public Property Let DatosEnv(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.DatosEnv = 5
   mvarDatosEnv = vData
End Property
Public Property Get DatosEnv() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.DatosEnv
   DatosEnv = mvarDatosEnv
End Property
Public Property Let TotEnv(ByVal vData As Integer)
' se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
' Syntax: X.TotEnv = 5
   mvarTotEnv = vData
End Property
Public Property Get TotEnv() As Integer
' se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
' Syntax: Debug.Print X.TotEnv
   TotEnv = mvarTotEnv
End Property
```

15 Módulo de clase *ResCan*.

```
Option Explicit

'   Este es el objeto ResCan, que almacena los resultados de un canal de la simulación
'   consta de cinco propiedades:
'      - MRoudTrip: retardo medio de vuelta
'      - MeanTrip: retardo medio
'      - Uso: uso del enlace
'      - Rend: rendimiento del protocolo
'      - Cef: cadencia eficaz

'  variables locales para almacenar los valores de las propiedadesPrivate mvarMeanTrip As Single '  copia local
Private mvarMRoundTrip As Single '  copia local
Private mvarMeanTrip As Single '  copia local
Private mvarUso As Single '  copia local
Private mvarRend As Single '  copia local
Private mvarCef As Single '  copia local

Public Property Let Cef(ByVal vData As Single)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Cef = 5
    mvarCef = vData
End Property


Public Property Get Cef() As Single
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Cef
    Cef = mvarCef
End Property



Public Property Let Rend(ByVal vData As Single)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Rend = 5
    mvarRend = vData
End Property


Public Property Get Rend() As Single
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Rend
    Rend = mvarRend
End Property



Public Property Let Uso(ByVal vData As Single)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.Uso = 5
    mvarUso = vData
End Property


Public Property Get Uso() As Single
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.Uso
    Uso = mvarUso
End Property



Public Property Let MRoundTrip(ByVal vData As Single)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.MRoundTrip = 5
    mvarMRoundTrip = vData
End Property


Public Property Get MRoundTrip() As Single
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.MRoundTrip
    MRoundTrip = mvarMRoundTrip
End Property



Public Property Let MeanTrip(ByVal vData As Single)
'  se usa al asignar un valor a la propiedad, en la parte izquierda de una asignación.
'  Syntax: X.MeanTrip = 5
    mvarMeanTrip = vData
End Property


Public Property Get MeanTrip() As Single
'  se usa al recuperar un valor de una propiedad, en la parte derecha de una asignación.
'  Syntax: Debug.Print X.MeanTrip
    MeanTrip = mvarMeanTrip
End Property
```