# Aplicación de prueba de los algoritmos

En este capítulo se expone el proceso seguido para el diseño e implementación de la aplicación utilizada para comprobar el correcto funcionamiento de los dos algoritmos. Principalmente se va a simular el comportamiento de una aplicación de paso de mensajes entre diferentes nodos de la red.

## Diseño

Como se ha mencionado anteriormente, la finalidad de la aplicación demostradora es la de simular el comportamiento de una aplicación que pueda tener una WSN. En concreto se busca reproducir el comportamiento de los nodos en los escenarios simulados de [refPaperJavi] y [refPaperElena].

Ya que el escenario presentado en [refPaperJavi] es bastante genérico, sin asumir ningún tiempo entre paquetes fijo y no podemos crear una red de sensores con muchos nodos, para el diseño de la aplicación asumiremos el escenario descrito en [refPaperElena]. En este escenario se suponen 100 nodos comunicándose en la banda de 2,4 GHz. Cada nodo envía un paquete de aplicación cada segundo.

//Pongo aquí una figura de un ejemplo de escenario típico???

Para el nuestra aplicación se va a hacer que los nodos envíen mensajes de aplicación cada segundo, variando aleatoriamente con un retardo de hasta cien milisegundos para comprobar el funcionamiento del algoritmo de seguridad. Se usarán dos nodos ya que son suficientes para comprobar la correcta implementación del código. Para el algoritmo de seguridad se van a implementar los siguientes cambios en tiempo de ejecución:

* Un nodo cambia su potencia transmitida en cierto momento, comprobando que el otro lo detecta como atacante.
* En otro momento el segundo nodo cambia el tiempo entre paquetes detectándolo el nodo anterior como atacante.

Para la prueba del funcionamiento del algoritmo de reducción de consumo es necesario que haya retransmisiones en el canal por el que se transmiten los mensajes de aplicación por tanto, es posible introducir una fuente de ruido en la frecuencia en la que se está transmitiendo para producir las retrasmisiones.

Debido a que van a existir mensajes de aplicación y de control es necesario definir una prioridad para el procesamiento de estos. Se han tomado las siguientes decisiones en cuanto al envío de mensajes de aplicación durante la ejecución de las estrategias:

* Seguridad: Al ser necesario un tiempo de procesamiento muy corto para la toma de decisiones, los mensajes de aplicación se siguen enviando normalmente. Cuando un mensaje de control llega, se procesa antes que todos los de aplicación que estén en el *buffer*.
* Reducción de consumo: Debido a que la decisión de cambiar de canal afecta al envío de mensajes de aplicación y que la decisión se toma teniendo en cuenta la opinión del resto de nodos de la red, se decide parar el envío de mensajes de aplicación durante la ejecución de la estrategia. El procesamiento de mensajes de aplicación sí sigue vigente durante la ejecución.

Con todo esto ya podemos pasar a describir la implementación de la aplicación de prueba.

## Implementación

La implementación de la aplicación tiene dos funciones que realizan las tareas necesarias y una más que se usa para procesar los mensajes recibidos. El código implementado se encuentra en el fichero Aplicacion.c. A continuación se detallan las funciones que contiene el fichero:

* void Enviar\_Paquete\_Datos\_App(radioInterface ri, BYTE modo, BYTE \*addr)

Esta función envía los mensajes de la aplicación. Para que se pare de enviar cuando se esté ejecutando el algoritmo de reducción de consumo se comprueba un flag siempre que se vaya a enviar un mensaje. También se encarga de contar el número de retransmisiones, guardarlo y de descartar los mensajes si se ha alcanzado el máximo de retransmisiones posibles. En los parámetros de entrada se le puede indicar la interfaz por la que se quiere transmitir, el modo de transmisión (*broadcast* o a un solo nodo) y en caso de que sea a un solo nodo, la dirección a la que se quiere enviar el mensaje.

* BOOL Proc\_Buff(RECEIVED\_MESSAGE \*Buffer)

Es la función que procesa los mensajes recibidos. Principalmente guarda la dirección del nodo que envió el paquete y los datos que contiene. También se encarga de dar prioridad a los mensajes de control haciendo que se procesen antes que los de aplicación si hay de los dos pendientes de procesar. Esto es sencillo ya que se ha reservado la interfaz de 434 MHz para hacer de VCC lo que permite comprobar simplemente si hay datos en el buffer de recepción de esta interfaz o en la que está transmitiendo la aplicación.

* void Recibir\_info(void)

Esta función recibe todos los mensajes y comprueba si son de control o de aplicación, almacenándolos o realizando las tareas necesarias para la aplicación. En este caso se ha decidido no hacer nada con los mensajes de aplicación ya que si se decidiese enviar por la UART los mensajes de aplicación recibidos se perderían las trazas más importantes de la ejecución de las estrategias.

Las decisiones tomadas en cuanto a la ejecución de las funciones de la aplicación para que se cumpla la temporización, en el caso del envío de mensajes, y no perder ningún paquete, en el caso de la recepción, han sido la de utilizar el *timer 5* que no estaba siendo utilizado para ejecutar la función de envío de mensajes. El *timer* ha sido configurado con un período de un milisegundo y prioridad inferior al resto de *timers*. Las líneas que configuran el *timer* 5 son las siguientes:

WORD TiempoT5 = 1; //En mseg

WORD CuentaT5 = (TiempoT5)\*(CLOCK\_FREQ/((1<<mOSCGetPBDIV())\*Prescaler\*1000));

OpenTimer5(T5\_ON | T1\_IDLE\_CON | T5\_GATE\_OFF | T5\_PS\_1\_32 | T4\_SOURCE\_INT, CuentaT5);

ConfigIntTimer5(T5\_INT\_ON | T5\_INT\_PRIOR\_1 | T5\_INT\_SUB\_PRIOR\_3);

Para que el periodo de transmisión de los paquetes sea mayor se ha definido una variable mseg que se aumenta con cada interrupción del *timer* y se ejecuta la función de envío de mensajes cuando se alcanza el valor de una constante definida y con nombre Periodo.

Para la recepción de mensajes se ha decidido utilizar el bucle principal del programa ya que no se necesita una temporización estricta y además, para no intentar procesar mensajes que no se han recibido, se comprueba que haya datos en algún buffer de recepción antes de procesarlos.

Con esto conseguimos que se procesen todos los mensajes que se reciban de forma rápida y conforme llegan los datos y se envíen los paquetes de datos con el período que se estime oportuno.

A continuación se presenta un diagrama con los pasos seguidos por la aplicación y las condiciones que se tienen que cumplir para que se ejecuten las funciones.

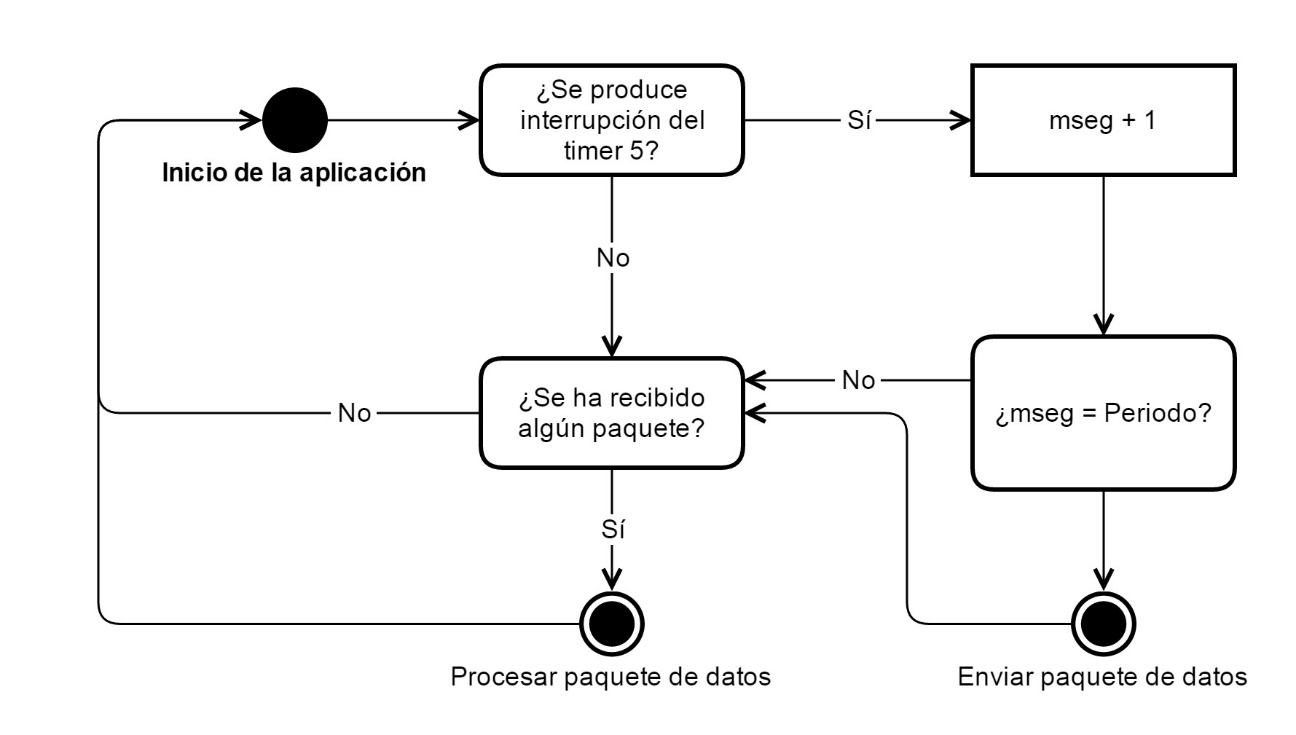


Figura 6.1 Diagrama de ejecución de la aplicación