# Implementación del algoritmo de reducción de consumo

En este capítulo se va a detallar el trabajo desarrollado para la implementación del algoritmo desarrollado en [refPaperElena]. Se explicarán las nuevas funciones añadidas en el CRModule y los mensajes y respuestas que se intercambian los nodos para conseguir el cambio a un canal más óptimo para la transmisión de mensajes de aplicación.

## Funciones de la arquitectura cognitiva

La implementación de este algoritmo se ha realizado siguiendo un esquema de máquinas de estados finitos tal y como se presenta en la Figura 5.1.

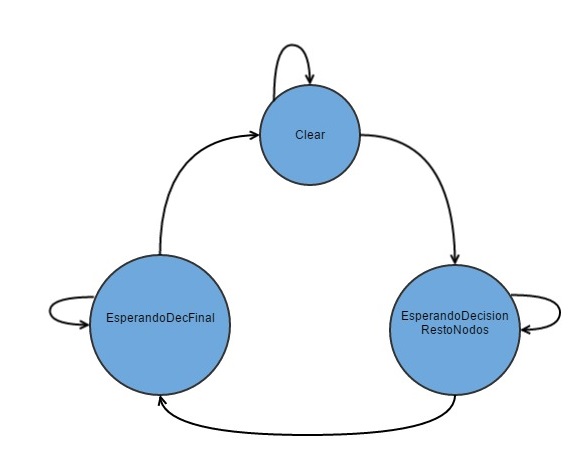


Figura 5.1 Diagrama de estados de la implementación propuesta

Las transiciones entre estados de la Figura 5.1 dependen de las respuestas de los otros nodos a las peticiones de cambio de canal que se les hace. Todos los nodos comenzarán en estado “Clear” en el que comprobarán, con cada mensaje de aplicación que envíen, el número de retransmisiones que se han realizado. Tras enviar el mensaje de aplicación y en la ejecución de la rutina de atención a la interrupción del *timer* 4, que se produce cada milisegundo, el nodo calcula los costes asociados a la transmisión en el canal en el que está transmitiendo y los costes asociados al cambio de canal. Cuando el coste de cambio es menor que el coste de transmitir en el canal actual, ese nodo inicia el proceso de elección del canal al que cambiar y notifica al resto de nodos en su red.

El paso de mensajes de petición de cambio de canal y las respuestas se producen a través de VCC, habiendo reservado la interfaz de 434 MHz disponible en los nodos para tal efecto.

A continuación se pasa a describir las funciones de cada uno de los sub-módulos del CRModule para procesar las peticiones de cambio y obtener el resultado final.

### Optimizer

El sub-módulo Optimizer es el encargado de la ejecución del algoritmo. Decidirá el inicio del cambio de canal, procesará las respuestas de los otros nodos de la red y pedirá al resto de sub-módulos la ejecución de determinadas acciones o la información que necesite durante el proceso. La estructura de los mensajes dirigidos a este sub-módulo y que se usará posteriormente para enviarle información desde otros sub-módulos es la siguiente:

typedef struct \_OPTM\_MSSG\_RCVD

{

INPUT BYTE OrgModule;

INPUT OPTACTION Action;

IOPUT void \*Param1;

IOPUT void \*Param2;

INPUT radioInterface Transceiver;

INPUT BYTE \*EUINodo;

} OPTM\_MSSG\_RCVD;

Las funciones que se han implementado o modificado en este sub-módulo se exponen a continuación:

* BOOL CRM\_Optm\_Cons(OPTM\_MSSG\_RCVD \*Peticion)

Es la función encargada de pedir al sub-módulo Discovery la información del ruido en los canales de las interfaces de 868 MHz y 2,4 GHz, obteniéndose así un canal óptimo por cada interfaz y el valor del RSSI del ruido en ese canal. De esos dos canales se elegirá el más óptimo y se guardará la interfaz a la que pertenece.

Una vez hecho esto se distingue entre si el algoritmo ha sido iniciado en el propio nodo, es decir, el parámetro Action del mensaje que se ha recibido era SubActCambio, y lo que hace es enviar un mensaje por VCC dirigido al sub-módulo Repository del resto de nodos con la información sensada. O por el contrario, lo que ha iniciado el algoritmo ha sido un mensaje de otro nodo, que lo que hace es procesar la información que se ha recibido, detectando si el canal que ha recibido del otro nodo estaba entre sus cuatro mejores (o con un valor de ruido muy próximo al de su cuarto mejor) y acepta el canal propuesto o le manda su mejor canal al resto de nodos. Esto último se realiza cuando el parámetro Action del mensaje que recibe es SubActRespCambio. Todo esto queda resumido en la Figura 5.2.

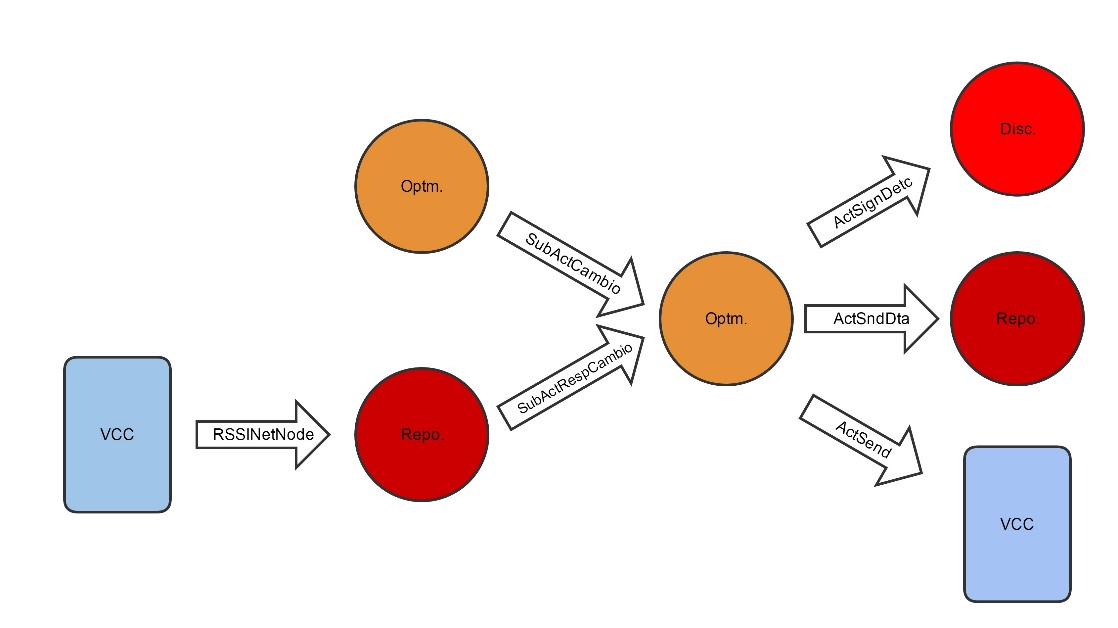


Figura 5.2 Diagrama con las acciones que realiza la función Cons y los parámetros Action que se pasan.

* BOOL CRM\_Optm\_Calcular\_Costes(BYTE n\_rtx)

Esta función se encarga de calcular los costes asociados a la transmisión en un canal ruidoso y los costes de cambiar de canal para decidir si iniciar o no el proceso para cambiar de canal. Se pide de argumento el número de retransmisiones del último mensaje de aplicación que se ha intentado enviar y el valor de salida es *TRUE* o *FALSE* dependiendo de la decisión de cambio de canal.

* BOOL CRM\_Optm\_Processor(OPTM\_MSSG\_RCVD \*Peticion)

Esta función es la que se encarga de procesar los mensajes provenientes de otros nodos. Existen tres tipos de mensajes de respuesta, una petición de cambio de canal, una respuesta a la petición de cambio o una decisión final.

Dependiendo del estado en el que se encuentre el nodo receptor y del mensaje que reciba se realizará una acción u otra. Las acciones realizadas se detallan a continuación:

* + Estado “Clear”. Cuando se esté en este estado sólo se procesarán los mensajes de petición de cambio de canal. El parámetro Action del mensaje dirigido al sub-módulo Optimizer tiene que ser ProcCambioCanal.

Cuando llegue un mensaje con ese parámetro se calculan los costes de cambio de canal y de transmisión en el canal actual y se manda respuesta al resto de nodos. Si la respuesta es positiva se llama a la función CRM\_Optm\_Cons para que se realice el sensado del medio se decida el canal óptimo. Todo esto queda resumido en la Figura 5.3.

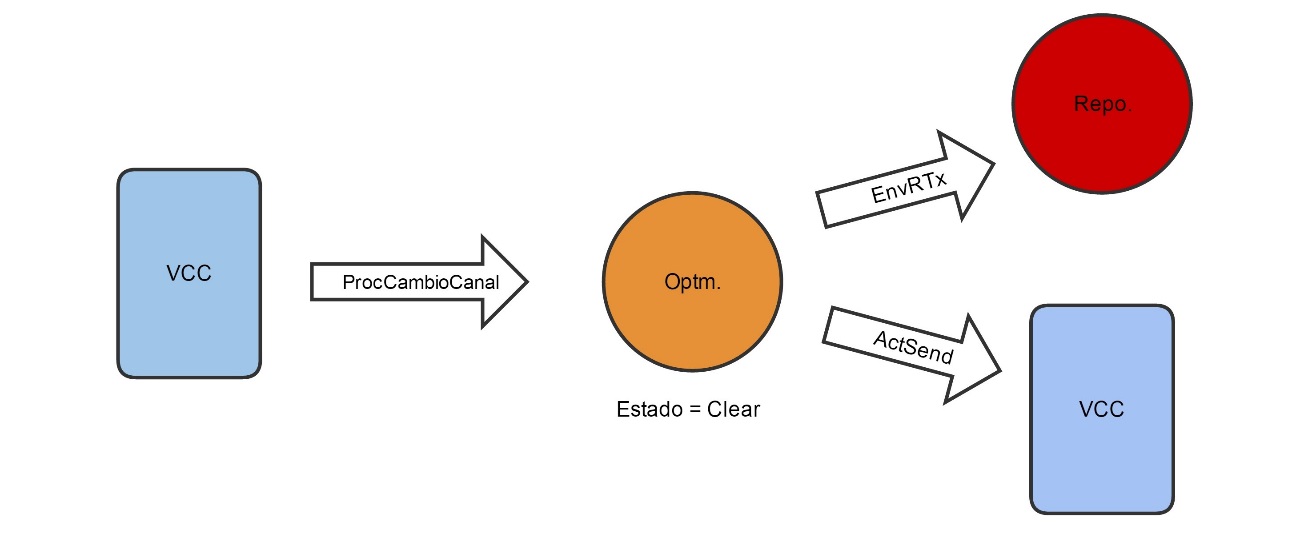


Figura 5.3 Diagrama de mensajes entre sub-módulos cuando se recibe una petición de cambio de canal

* + Estado “EsperandoDecisionRestoNodos”. En este estado se esperan las respuestas al cambio de canal de todos los nodos que estén conectados. Si se recibe una respuesta negativa se evalúa el porcentaje de mensajes de aplicación que se han intercambiado los dos y si hay uno con el que se comunica mucho se rechaza el cambio de canal; si no se comunica mucho sigue con el proceso de cambio. El parámetro Action del mensaje dirigido al sub-módulo Optimizer tiene que ser ProcCambioCanal.

Los pasos de mensajes entre los sub-módulos cuando se recibe una respuesta a un mensaje de cambio de canal queda resumido en la Figura 5.4.

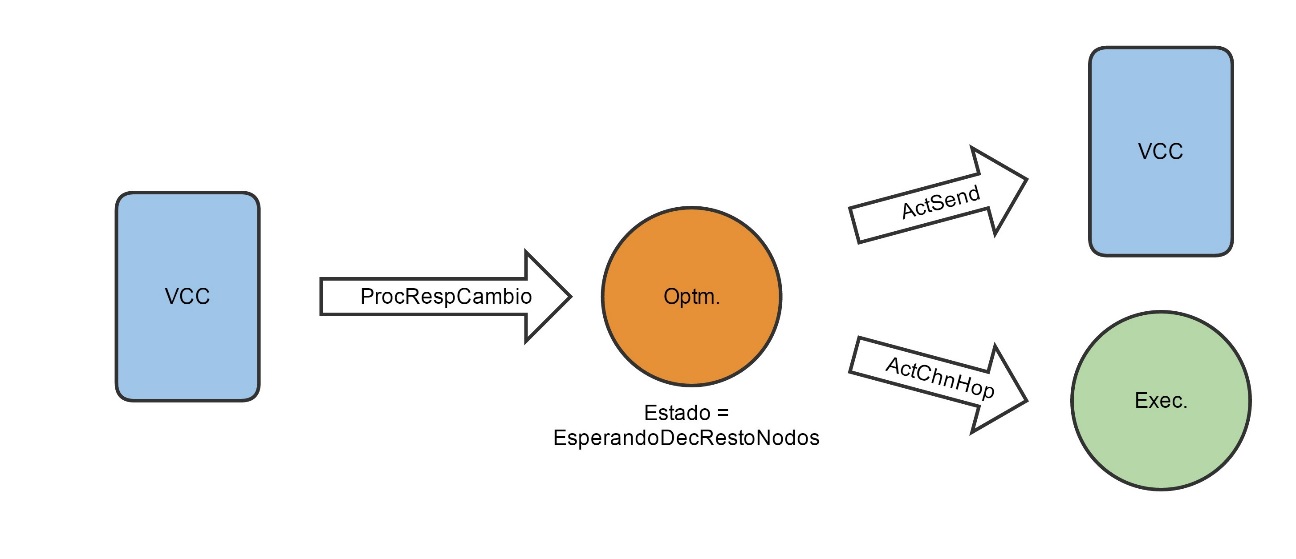


Figura 5.4 Diagrama de mensajes entre sub-módulos cuando se recibe una respuesta a una petición de cambio de canal

* + Estado “EsperandoDecFinal”. En este estado todos los nodos en la red deciden a qué canal cambiarse. Cada nodo recibe la información del espectro del resto de nodos y comprueba si el canal elegido está entre sus cuatro mejores. Si está entre los cuatro manda un mensaje al resto de nodos confirmando que se va a cambiar y se cambia de canal. Si no está entre esos cuatro comprueba si el nivel de ruido del cuarto mejor canal es parecido al del canal que le han enviado y si lo es acepta el cambio. El resultado de este estado siempre va a ser un cambio de canal. El parámetro Action del mensaje dirigido al sub-módulo Optimizer tiene que ser ProcRespCambio si es un mensaje de respuesta afirmativa o negativa a una petición, o ProcDecFinal si ya se ha recibido el canal óptimo de ese nodo y se ha decidido no cambiar a ese. Los pasos de mensajes entre sub-módulos cuando se recibe una respuesta con un canal distinto al que se propuso inicialmente viene resumido en la Figura 5.5.

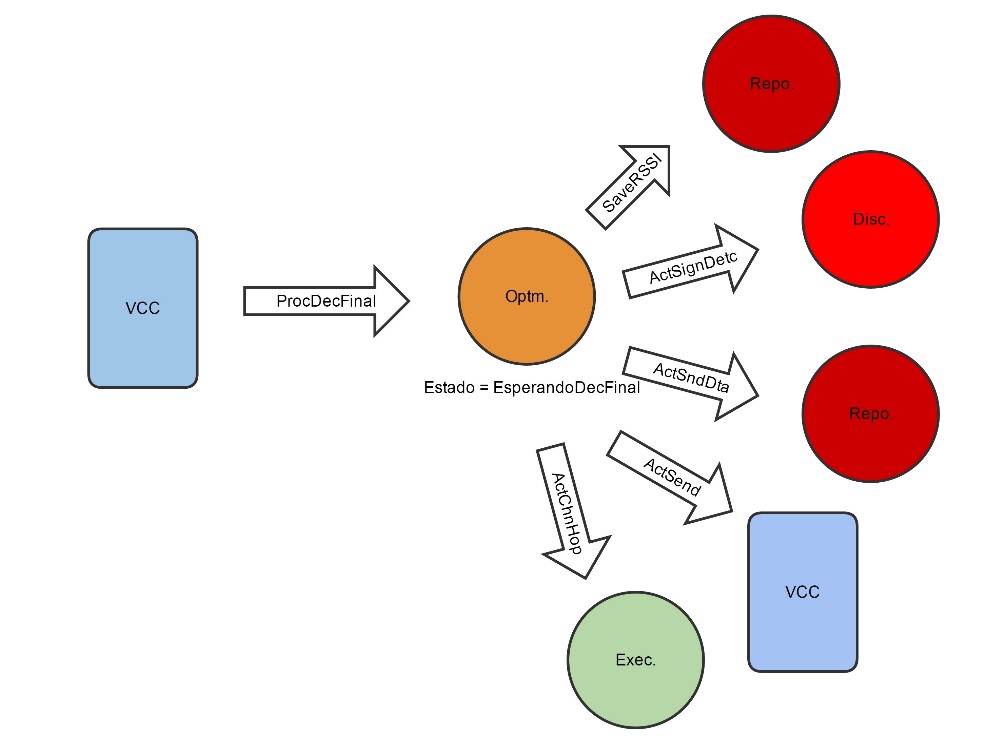


Figura 5.5 Diagrama de mensajes entre sub-módulos cuando se recibe una respuesta de cambio de canal con un canal diferente al propuesto inicialmente

### Repository

Este sub-módulo será el encargado de almacenar la información que se reciba del resto de nodos de la red y enviarla al resto de sub-módulos si se la piden. La estructura de los mensajes con destino este sub-módulo es la siguiente:

typedef struct \_REPO\_MSSG\_RCVD

{

INPUT BYTE OrgModule;

REPACTION Action;

INPUT BYTE \*EUINodo;

INPUT REPODATATYPE DataType;

INPUT radioInterface Transceiver;

IOPUT void \*Param1;

IOPUT void \*Param2;

IOPUT void \*Param3;

IOPUT void \*Param4;

} REPO\_MSSG\_RCVD;

Cuando le lleguen mensajes de VCC los datos se almacenarán en Param2 como se explicará en la siguiente sección y cuando le lleguen mensajes de Optimizer devolverá la información en diferentes parámetros dependiendo del dato pedido. La información que se puede pedir y almacenar en Repository y las funciones implementadas se detallan a continuación:

* BOOL CRM\_Repo\_SendDat(REPO\_MSSG\_RCVD \*Peticion)

Esta función es la encargada de enviar la información que se le pide al sub-módulo Repository desde otros sub-módulos. Para la implementación de este algoritmo se ha añadido la posibilidad de devolver el número de retransmisiones en un canal, el canal y el RSSI en una posición de la lista de canales ordenada y el número de mensajes intercambiados con otro nodo.

Para devolver el canal y el RSSI de una posición, se ordena la lista de RSSI de menor a mayor, ocupando la menor potencia la posición cero, y ordenando una lista con los números de canal de la misma forma que la de RSSI. El algoritmo de ordenación usado para esto ha sido *Bubble Sort.*

* void CRM\_Repo\_NodoPropio(REPO\_MSSG\_RCVD \*Peticion)

En esta función se ha añadido la opción de almacenar el número de retransmisiones de un paquete de aplicación mediante el envío de un mensaje a Repository con el campo Param1 del mensaje con el valor EnvRTx.

* BOOL CRM\_Repo\_NodosRed(REPO\_MSSG\_RCVD \*Peticion)

Esta función se encarga de recibir los mensajes de otros nodos con información de sensado del otro nodo. Almacena el valor del canal óptimo y de su potencia de ruido para las interfaces de 868 MHz y 2,4 GHz además del mejor canal de entre esos dos y la interfaz a la que pertenece. Dependiendo del estado en el que se encuentre el nodo en el momento en que recibe el mensaje, manda una petición al método CRM\_Optm\_Processor con diferente Param1 para que el sub-módulo Optimizer procese la información del mensaje.

El paso de mensajes entre Repository y Optimizer cuando recibe información del espectro de otros nodos y el campo Param1 para cada estado queda detallado en la Figura 5.6.

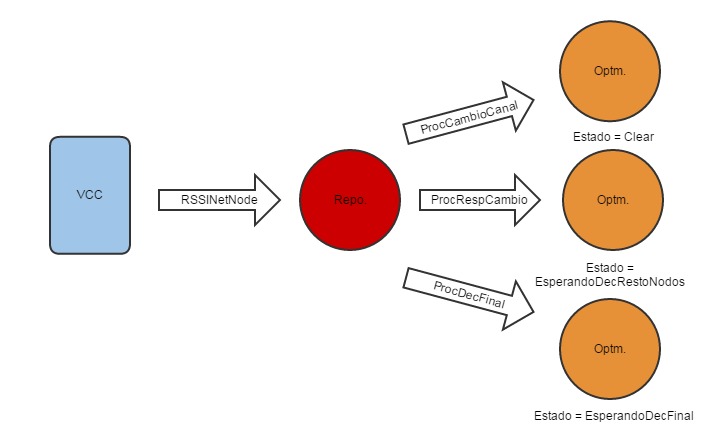


Figura 5.6 Diagrama de peticiones a Optimizer cuando se recibe información de sensado de otro nodo

* void CRM\_Repo\_NRTx(BYTE n\_rtx, BYTE canal, radioInterface ri)

Es la función encargada de almacenar el número de retransmisiones que se producen al enviar un mensaje de aplicación. Los parámetros de entrada son la interfaz radio con la que se está transmitiendo, el canal en el que se han producido las retransmisiones y el número total de retransmisiones que se han producido.

* void CRM\_Repo\_Mensajes\_Intercambiados(BYTE \*Address)

Esta función es la encargada de actualizar el número de mensajes que se han recibido de un mismo nodo durante la ejecución de la aplicación. La llamada a esta función se realiza mediante un mensaje a Repository con el campo DataType con el valor AddMsg. El mensaje se envía cuando en la interrupción del timer 4 se comprueba si hay datos en el buffer de recepción.

* void CRM\_Repo\_Str\_RSSI(radioInterface ri)

Esta función se encarga de guardar el valor de la potencia de ruido térmico obtenida del sensado del espectro realizado anteriormente por el sub-módulo Discovery. Posteriormente se utiliza esta información para decidir si el canal al que quiere cambiar otro nodo de la red es adecuado para el nodo que hace el sensado.

* BOOL CRM\_Repo\_Reiniciar\_RTx(void)

Es una función auxiliar. Se llama a esta función cada vez que se decide no cambiar de canal debido a que se decide no cambiar de canal porque un nodo con el que se ha intercambiado un porcentaje elevado de mensajes ha decidido no cambiar de canal. Poniendo a cero el número de retransmisiones producidas se evita que se inicie el algoritmo dos veces consecutivas sin que se intente enviar un mensaje de aplicación nuevo.

### VCC

En este sub-módulo se ha tenido que modificar el modo en el que se pasaban los mensajes recibidos con destino Repository. Al tener que enviar un número de datos superior al número de campos en el mensaje con posibilidad de incluir datos personalizados, y al disponer de punteros con la capacidad de incluir cualquier tipo de dato, se ha procesado la información útil de cada mensaje, incluyéndolo en un vector y pasando ese vector a través del puntero al sub-módulo destino.

### Execution

La funcionalidad de este sub-módulo no ha sido modificada. Las peticiones que se le hacen desde Optimizer son para cambiar el canal en el que se está transmitiendo.

### Discovery

La funcionalidad de este sub-módulo tampoco ha sido modificada. Los mensajes que le envía Optimizer son para pedir la información del canal óptimo de cada interfaz y el valor de ruido térmico que reciben.

## Comentarios

La funcionalidad de enviar mensajes *broadcast* mediante el sub-módulo VCC no estaba implementada en la HAL ni en el sub-módulo por lo que se ha modificado la estructura de los mensajes que se envían a VCC añadiendo el parámetro AddrMode. Además se ha modificado la función Send\_Buffer en la HAL añadiendo un parámetro de entrada con el mismo nombre. Con esto, pasando el parámetro BROADCAST\_ADDRMODE a los mensajes dirigidos a VCC se consigue la nueva funcionalidad.