3. MENSAJES

Los objetos *Message* son, en la simulación, la base para la transferencia de información entre nodos BGP. Realmente en el simulador la base para la transferencia de información son los objetos *Packet*, pero un nodo BGP no es capaz de interpretar su contenido. Podemos ver la clase *Message* como la PDU del entorno BGP y la SDU de la base de simulación; a su vez, podemos ver la clase *Packet* como la PDU de la base de simulación.

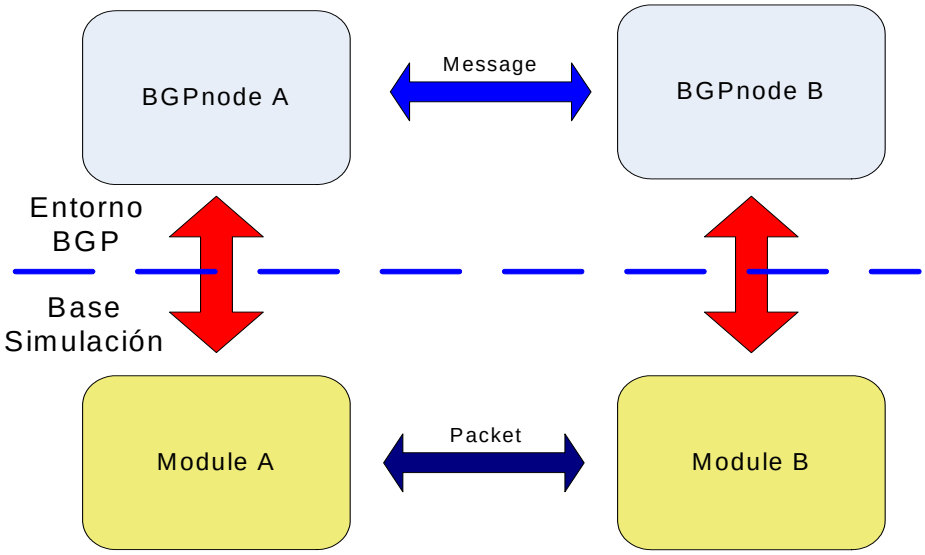

Figura 2. Modelo de transmisión de información.

La clase *Message* deriva de la clase *Packet*, y las clases para mensajes específicos (*MessageOpen*, *MessageNotif* y *MessageUpdate*) derivan a su vez de la clase *Message*.

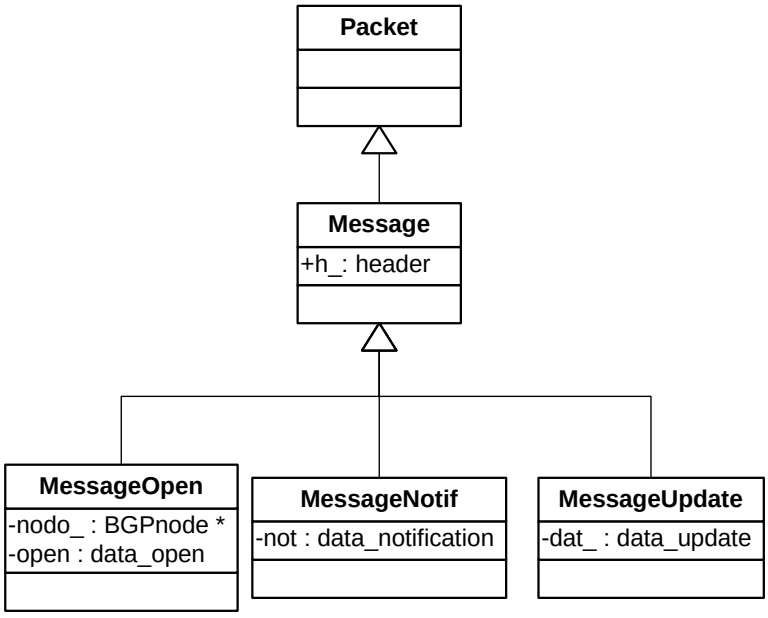

Figura 3. Diagrama de clases UML de Message.

Para cada una de las clases asociadas a los mensajes BGP sólo se han definido un constructor por defecto (sin parámetros) y otro con paso de parámetros.

3.1. LA CLASE *Message*

Como se ha dicho, esta clase representa la clase base de los mensajes BGP. Debido a que el intercambio de información en el simulador (librería *simul*) está basado en la clase *Packet*, se ha tomado esta última como clase padre de *Message* para poder intercambiar mensajes entre nodos BGP (usando *simul*).

La clase *Message* tiene dos funcionalidades; la primera de ellas es ser la clase base para las clases que representan los mensajes OPEN, NOTIFICATION y UPDATE que se intercambian entre nodos BGP vecinos y la segunda es la de ser utilizada para el intercambio de mensajes KEEPALIVE, esto se hace de así porque la clase *Message* sólo tiene definida una variable miembro, que es de tipo *header* y representa a una cabecera BGP (cuya estructura es común a todos los tipos de mensajes), y los mensajes KEEPALIVE sólo contienen la cabecera.

3.1.1. Variables miembro

header h_

Se trata de una variable de tipo *header* (definida en *libr.h*) y representa la cabecera de un mensaje BGP.

3.1.2. Métodos

Message::Message()

Se trata de un constructor por defecto de la clase *Message* y es llamado por el constructor de la clase *gestConn* para inicializar su variable miembro *mm*.

En este constructor se genera una estructura de tipo *header* (*hdefault*) que se almacenará en *h_*. Los valores asignados serán 0x00000000 para los campos *marker1*, *marker2*, *marker3* y *marker4*, y 0 para los campos *length* y *type*.

Message::Message(header he, double size)

El constructor de la clase *Message* recibe dos parámetros, el primero de ellos es de tipo *header* y se almacenará en la variable miembro *h_*. El segundo de estos parámetros es un *double* que indica el tamaño, en bytes, del mensaje y que se emplea para pasar como argumento al constructor de la clase padre de *Message* (*Packet*).

3.2. LA CLASE MessageOpen

Esta clase representa a los mensajes OPEN. Como se ha mencionado anteriormente, deriva de la clase *Message* y de ésta hereda la variable *h_*, que representa la cabecera del mensaje.

3.2.1. Variables Miembro

BGPnode * nodo_

Se trata de una referencia al nodo que genera el mensaje OPEN. Se utiliza para poder acceder a los valores del nodo necesarios para rellenar los campos de la variable miembro *open* (identificador de AS, valor del *Hold Timer* y dirección IP del nodo transmisor).

data_open open

Se trata de una estructura *data_open* (definida en *libr.h*). En ella se almacenan los datos concernientes al mensaje OPEN, es decir, se puede ver como el campo de datos de los mensajes OPEN.

Esta variable se ha declarado como *private*, por lo que tiene el acceso restringido a aquellos métodos que no sean de la misma clase. No obstante, se ha declarado como *friend* al método *BGPnode::analyseOpen(…)* que es el método de la clase *BGPnode* que se encarga del análisis de los mensajes OPEN.

3.2.2. Métodos

MessageOpen::MessageOpen()

Se trata del constructor por defecto de la clase *MessageOpen* y es llamado por el constructor de la clase *gestConn* para inicializar su variable miembro *mopen*.

En este constructor se genera una estructura de tipo *header* (*hdefault*) que se almacenará en *h_* (heredada de *Message*). Los valores asignados serán 0x00000000 para los campos *marker1*, *marker2*, *marker3* y *marker4*, 1 para el campo *type* y 0 para el campo *length*. A la variable *nodo_* se le asigna el valor *null*. Y finalmente se le asignan valores por defecto a los campos de la variable *open* (*version*=4, *bgpip*=0.0.0.0, *my_as*=0, *hold_time*=0 y *opl*=0).

MessageOpen::MessageOpen(BGPnode * nodo, header h, double size, unsigned short int h_time)

Se trata del constructor empleado para generar un mensaje OPEN que seguidamente será enviado. Como vemos, se le pasan cuatro parámetros. El primero de ellos es una referencia a un objeto *BGPnode*, esta referencia se almacena en *nodo_* para poder completar los valores necesarios en la variable miembro *open* (*version*, *my_as*, *bgpid* y *opl*). El segundo es una estructura *header*, que se almacena en *h_*. A su vez la estructura *header* junto con el valor *double* se emplearán para la llamada al constructor de la clase padre *Message*. Finalmente el parámetro *h_time* se emplea para dar valor al campo *open.hold_time*, este último valor no se toma directamente del nodo, sino del gestor de conexión, es por esto por lo que se le pasa este valor concretamente y no a través de la referencia del nodo BGP. La ejecución del constructor consiste en la asignación de estos valores.

3.3. LA CLASE MessageNotif

La clase *MessageNotif* representa a los mensajes NOTIFICATION. La

recepción o el envío de este tipo de mensajes supone el cese de la comunicación. Al igual que la clase *MessageOpen*, deriva de la clase *Message* y, por tanto hereda la variable miembro *h_*.

3.3.1. Variables Miembro

data_notification not

Se trata de una estructura de tipo *data_notification* (definida en *libr.h*). Se emplea para almacenar los datos relativos a un mensaje NOTIFICATION.

Esta variable se ha declarado como *private*, por lo que tiene el acceso restringido a aquellos métodos que no sean de la misma clase. No obstante, se ha declarado como *friend* al método *BGPnode::analyseNotif(…)* que es el método de la clase *BGPnode* que se encarga del análisis de los mensajes NOTIFICATION.

3.3.2. Métodos

MessageNotif::MessageNotif()

Este es el constructor por defecto de la clase *MessageNotif* y lo llama el constructor de la clase *gestConn* para inicializar su variable miembro *mnotif*.

En este constructor se genera una estructura de tipo *header* (*hdefault*) que se almacenará en *h_* (heredada de *Message*). Los valores asignados serán 0x00000000 para los campos *marker1*, *marker2*, *marker3* y *marker4*, 2 para el campo *type* y 0 para el campo *length*. A los distintos campos de la variable miembro *not* se les asigna valores por defecto (*error*=0, *err_subcode*=0, *p_data*=NULL).

MessageNotif::MessageNotif(header h, char err, char err_subc, void * pd, double size)

Se trata del constructor empleado para generar un mensaje NOTIFICATION que seguidamente será enviado. Como se puede observar se le pasan cinco valores a este constructor; el primero de ellos es una estructura de tipo *header* que se asigna a *h_*. El segundo de ellos es un *char* que indica el código de error y que se asigna a *not.error*. El siguiente parámetro es el subcódigo de error, que se asigna a *not.err_subcode*. Le sigue un puntero a *void* que se asigna a *not.p_data*. Y finalmente tenemos un *double* (*size*), que indica la longitud en bytes del mensaje y que, como en ocasiones anteriores, se emplea junto con la estructura *header* para invocar al constructor de la clase padre.

3.4. LA CLASE MessageUpdate

La clase *MessageUpdate* representa a los mensajes UPDATE; éstos se emplean para indicar la presencia de nuevas rutas o bien que rutas que eran factibles han dejado de serlo. Esta clase deriva de la clase *Message* y, como en los casos anteriores, hereda la variable miembro *h_*.

3.4.1. Variables Miembro

data_update dat_

Ésta es una variable de tipo *data_update* cuya misión es la de almacenar los datos de un mensaje UPDATE.

Esta variable ha sido declarada como *private*, por lo que tiene el acceso restringido a aquellos métodos que no sean de la misma clase. No obstante, se ha declarado como *friend* al método *BGPnode::analyseUpd(…)* que es el método de la clase *BGPnode* que se encarga del análisis de los mensajes UPDATE.

3.4.2. Métodos

MessageUpdate::MessageUpdate()

Se trata del constructor por defecto de la clase *MessageUpdate* y lo llama el constructor de la clase *gestConn* para inicializar su variable miembro *mupd*.

En este constructor se genera una estructura de tipo *header* (*hdefault*) que se almacenará en *h_* (heredada de *Message*). Los valores asignados serán 0x00000000 para los campos *marker1*, *marker2*, *marker3* y *marker4*, 4 para el campo *type* y 0 para el campo *length*. A los distintos campos de la variable miembro *dat_* se les asigna valores por defecto (*unf_route_l*=0, *tot_PA_l*=0, *announced.prefix*=0.0.0.0, *announced.length*=24).

MessageUpdate::MessageUpdate(header h, data_update * upd, double size)

Este es el constructor empleado para generar un mensaje UPDATE que seguidamente será enviado. Los parámetros que se le pasan son: una estructura de tipo *header* que se almacena en *h_*, una referencia a una estructura de tipo *data_update* que servirá para asignar valores a los campos de *dat_* y un *double* (*size*) que indica el tamaño del mensaje en bytes y que se emplea con la estructura *header* para invocar al constructor de la clase padre de *MessageUpdate*.

En este constructor hay una diferencia notable respecto a los constructores de las clases *MessageOpen* y *MessageUpdate*, y es que en estos últimos el campo de datos del mensaje (*data_open* y *data_notification* respèctivamente) se va generando en el constructor, bien a partir de una referencia al nodo (*MessageOpen*) o bien se le pasan directamente los valores a asignar (*MessageNotif*); pero en el caso de este constructor el campo de datos lo genera quien lo invoca y le pasa una referencia a este constructor al invocarlo. Esto se hace así por simplicidad, ya que si no habría que pasarle todos y cada uno de los campos de la estructura *data_update*, haciendo inecesariamente engorroso el código.