2. LIBRERÍA

En el archivo de cabecera *libr.h* se han introducido las estructuras de datos que serán usadas durante la simulación. En todo momento se ha tratado de respetar las estructuras, tanto en tamaño como en orden y parámetros, descritas por la **RFC 1771**, aunque, obviamente, en casos muy concretos y puesto que no aportaban a la simulación se ha prescindido de esta "regla", ya que no debemos olvidar que se trata de una simulación y no de un sistema real.

A continuación se pasará a la descripción de los distintos elementos que componen este archivo.

2.1. DIRECCIONES IP

Para las direcciones IP se ha utilizado un *struct* con el nombre *ip*. Este *struct* consta de cuatro campos, que lógicamente se corresponden con los cuatro bytes que constituyen una dirección IPv4. Como ya se ha indicado, se pretende ser lo más fiel a la norma, con lo que para conseguir estos bytes se ha empleado variables de tipo *unsigned char*. El tipo *char* porque ocupa 8 bits (1 byte) y *unsigned* porque las direcciones IP no contemplan números negativos (de 0 a 255). Los distintos campos se han nombrado: *a*, *b*, *c* y *d*; siendo *a* el primer byte y *d* el último byte de la dirección IP.

2.2. MENSAJES

Del estudio de la **RFC 1771** se deriva que todos los tipos de mensaje constan de dos partes claramente diferenciadas. La primera de ellas es la cabecera, que en cuanto a estructura es común a todos los mensajes, y en segundo lugar los datos del mensaje en sí, que son característicos e inherentes a cada tipo de mensaje.

2.2.1. Cabecera

Como ya se ha dicho, la cabecera es, en cuanto a su estructura, común a todos los mensajes, por lo que se ha definido un *struct*, llamado *header*, que se usará en la creación de todos los mensajes. La **RFC 1771** describe los distintos campos de la cabecera, conformando un total de 19 bytes. El primero de estos campos es el *Marker*, que consta de 16 octetos, estos 16 octetos se han descompuesto en cuatro campos de bits (*bitfields)* de 4 bytes (32 bits) cada uno de ellos. Los valores de estos campos se explicarán en la descripción de la clase *BGPnode*. Su funcionalidad es comprobar la autenticidad del mensaje así como la posible pérdida de sincronismo. Después de éste viene el campo *length* que consta de 2 bytes y que indica la longitud del mensaje. Para su implementación se ha usado una variable de tipo *unsigned short int*, ya que se tratará de número entero no negativo; el hecho de que sea *short* es precisamente para que tan solo ocupe 2 bytes. Finalmente se encuentra el campo *type,* este campo consta de un solo byte e indica el tipo de mensaje de que se trata. Para su implementación se ha empleado una variable de tipo *char* (1 byte) y tomará los valores según el tipo de mensaje que se envíe: 1-OPEN, 2-UPDATE, 3-NOTIFICATION y 4-KEEPALIVE. Este campo nos servirá en la recepción para discernir qué tipo de mensaje se trata y así poder llamar al método apropiado para su

procesado.

2.2.2. Datos de Mensajes

Según el tipo de mensaje se han definido distintas estructuras de datos.

2.2.2.1. Mensaje OPEN

El *struct* definido para los mensajes OPEN es *data_open*. El primer elemento en esta estructura de datos es *version*, se trata de una variable de tipo *char* debido a que ocupa un único byte; este campo indica la versión de BGP empleada, por tanto, su valor será siempre "4" (BGP v4). Este campo podría ser prescindible, pero se ha mantenido por fidelidad a la norma.

El segundo campo en la estructura de datos es *my_as* (My Autonomous System), se trata de una variable de tipo *unsigned short int*, *unsigned* porque no tiene sentido el uso de identificadores de sistema autónomo negativos, y *short int* para que ocupe 2 bytes (tomará valores entre 0 y 65535), este campo indica el sistema autónomo al que pertenece el nodo que envía el mensaje OPEN. El valor de este campo será constante para todos los mensajes OPEN de un nodo concreto.

El tercer campo es *hold_time*, se trata de una variable de tipo *unsigned short int*, *unsigned* porque se refleja un valor de tiempo, y *short int* para que ocupe 2 bytes. Este valor es la proposición, por parte de quien lo envía, para el valor del *Hold Timer* (en segundos), este valor es negociado y permanece el que sea menor. El valor de este campo será constante para cada nodo.

El cuarto valor de la estructura de datos, *bgpid*, es la dirección IP del nodo transmisor.

El último valor de la estructura de datos es *opl* (Optional Parameters Length), se trata de una variable de tipo *char* que indica la longitud (en bytes) de los parámetros opcionales. Su valor será siempre "0", ya que no se incluyen parámetros opcionales. La **RFC 1771** define como parámetros opcionales información de autenticación, pero se ha prescindido de esta información por simplicidad, ya que habría que introducir mecanismos de autenticación, y el objetivo del presente simulador no es probar mecanismos de autenticación, sino el funcionamiento de BGP (no obstante, en futuras ampliaciones se podrían introducir).

2.2.2.2. Mensaje NOTIFICATION

El *struct* definido para los mensajes NOTIFICATION es *data_notification*. El primer campo de la estructura de datos es *error* (Error Code), se trata de una variable de tipo *char* (ya que sólo ocupa 1 byte); su valor indica a quién lo recibe el tipo de error que se ha producido. La **RFC 1771** define los valores que tomará según el caso.

El segundo campo es *err_subcode* (Error Subcode); al igual que el primer campo, se trata de una variable de tipo *char*. Su finalidad es aportar información más detallada sobre el error producido. También sus valores están definidos por la norma.

El último campo de la estructura de datos es *p_data*, se trata de un puntero a *void*, para poder asignarle cualquier dato que se considere oportuno para ampliar aún más la información sobre el error producido. En principio se le asigna el valor *NULL*, ya

que no se considera que sea necesaria para la simulación más información que el código y subcódigo de error.

2.2.2.3. Mensaje KEEPALIVE

Para este tipo de mensajes no se ha definido ninguna estructura de datos, ya que la **RFC 1771** establece que los mensajes KEEPALIVE no llevan datos asociados. Por tanto, estos mensajes sólo tendrán la cabecera.

2.2.2.4. Mensajes UPDATE

Este tipo de mensajes es el más complejo, ya que como se verá más adelante consta de *structs* anidados y listas.

El *struct* que conforma el campo de datos de los mensajes UPDATE se llama *data_update*. Esta estructura de datos consta de cinco campos. El primero de ellos es *unf_route_length* (Unfeasible Routes Length), que es una variable de tipo *unsigned short int*, *unsigned* porque al tratarse de una longitud, no tiene sentido que sea negativa, y *short int* para que ocupe 2 bytes. El objetivo de este parámetro es indicar la longitud (en bytes) del campo que indica las rutas que dejan de ser válidas.

El segundo campo de la estructura de datos se llama *withdr_routes*, este campo consiste en una lista de estructuras *nlri* (se verán más adelante), donde en cada uno de los elementos de esta lista se anuncia una ruta que deja de ser válida (unfeasible). Se ha elegido una lista para representar este campo por varios motivos; en primer lugar porque se trata de una estructura de tamaño variable, esto supone una ventaja respecto a los *arrays*, ya que podemos albergar tantas rutas no válidas en un mensaje UPDATE como deseemos sin estar limitados a un número de elementos prefijados (nos puede hacer falta más elementos de lo que defina el tamaño del *array*, o bien podemos necesitar menos, esto en un *array* supone un desperdicio de memoria). En segundo lugar se ha elegido una lista en vez de un vector porque la lista exige un acceso ordenado, lo cuál es más lógico en este caso, ya que el receptor accederá a los distintos miembros para eliminarlos de sus tablas de enrutamiento y en esta situación tan crítica es mejor tener un acceso ordenado, cerrando la posibilidad a un acceso desordenado (lo cual es posible en vectores).

El tercer campo en la estructura de datos es *tot_PA_l* (Total Path Attributes Length), como en campos anteriores que indican una determinada longitud, se trata de una variable de tipo *unsigned short int*, los motivos de que sea así son los mismos que en el caso de *unf_route_l*. El objetivo de este parámetro es indicar la longitud total de los atributos de la ruta válida anunciada.

El siguiente campo en la estructura de datos es también una estructura. Este campo es *pa* (Path Attributes), este campo describe los atributos de la ruta anunciada como válida en el mensaje UPDATE. La estructura en sí se describe más adelante.

El último campo de la estructura de datos propia de los mensajes UPDATE es también una estructura (que también se explicará más adelante). Este campo se llama *announced* y consiste en la ruta válida anunciada por el nodo BGP transmisor a sus pares.

2.3. STRUCTS AUXILIARES

En el apartado anterior se ha mencionado la existencia de estructuras de datos que conforman campos de los mensajes UPDATE. Estas estructuras de datos son *nlri* y *path_att*.

El *struct nlri* es, como su propio nombre indica un NLRI (Network Layer Reachability Information), esta estructura consta de una dirección IP (el *struct ip* descrito previamente) y un campo que indica la longitud de bits a '1', desde el más significativo (desde la izquierda), que se usan para la máscara de subred. Por defecto se ha tomado una longitud de 24 bits, lo cual equivale a una máscara de subred 255.255.255.0.

El *struct path_att* agrupa los atributos de la ruta anunciada como válida en los mensajes UPDATE. En esta estructura se han suprimido campos descritos por la **RFC 1771** que se ha considerado poco útiles para la simulación o bien porque complican en exceso la simulación. Esta estructura de datos se compone de los siguientes campos: en primer lugar está *origin*, este campo indica el origen de la ruta, tomará los valores 0 (si es de origen local al nodo BGP considerado) y 2 (si es de origen externo al nodo BGP considerado); este campo es de tipo *char* para que ocupe un solo byte. El siguiente campo es una lista, *as_path*, y consiste en la secuencia de Sistemas Autónomos que ha ido atravesando el mensaje UPDATE desde su origen. El siguiente campo es *next_hop*, que consiste en una dirección IP (la dirección IP a la que hay enviar un mensaje de datos para que llegue hasta su destino). El último campo es *local_pref*, este campo indica la preferencia del nodo transmisor por la ruta que está siendo anunciada, es una variable de tipo *unsigned int* ya que debe ser un número entero no negativo. La preferencia máxima es 65535 y se da en el caso de que un nodo anuncie su propio AS. A medida que va pasando por distintos nodos se va decrementando.

2.4. RIBs

El *struct* RIB describe una entrada en las Bases de Información de Enrutamiento (conocidas como RIBs), que son: Adj_RIB_In, Loc_RIB y Adj_RIBs_Out. Cada una de éstas será un vector de *structs RIB*. Esta estructura de datos es básica para el correcto funcionamiento del protocolo BGP y, por tanto, básica para la simulación. Sus campos se describen ma continuación.

El primero de los campos es *stored*, este campo es el NLRI de un AS concreto, es decir, su dirección IP de entrada y la longitud de los bits de máscara de subred a considerar. El siguiente campo es *pta*, este campo es un *struct* path_att y, por tanto, consiste en el conjunto de atributos que describen el camino hasta el AS anunciado. El último campo es *idcn*, este campo indica la conexión por la que el nodo BGP ha conocido la existencia de esta ruta. En el caso de que el origen de la ruta sea local, este campo valdrá 0.