
Table of Contents

Introduction	1.1
OSPF	1.2
El protocolo OSPF	1.2.1
OSPF en Quagga	1.2.2
Laboratorio de OSPF	1.2.3
BGP	1.3
BGP en Quagga	1.3.1
Laboratorio de BGP	1.3.2
Control de tráfico	1.4
Control de tráfico en Linux	1.4.1
Laboratorio de Control de Tráfico	1.4.2
Redes privadas virtuales	1.5
Redes privadas virtuales con IPsec	1.5.1
Laboratorio de VPN	1.5.2

Introducción

Éste es un conjunto de prácticas de laboratorio para la configuración de protocolos de red.

Estas prácticas están pensadas para realizarlas en máquinas Linux. Para facilitar la prueba de estas prácticas, se utilizará el entorno de emulación de redes [NetGUI](#).

[Para conocer mejor NetGUI](#).

Licencia

BY-NC-SA Creative Commons

OSPF (Open Shortest Path First)

- Introducción
- Funcionamiento básico de OSPF
- Protocolo HELLO
 - DR
 - BDR
 - Ejemplo de elección de DR y BDR
- LS Update
 - Router LSA
 - Network LSA
 - Número de secuencia de un LSA
- Encaminamiento por inundación
 - Fiabilidad en la inundación de LSAs
- Bases de datos de OSPF
- Caducidad de los mensajes LSU
- Mensajes entre diferentes áreas OSPF
 - Summary-LSA
- Resumen de mensajes OSPF
- Referencias

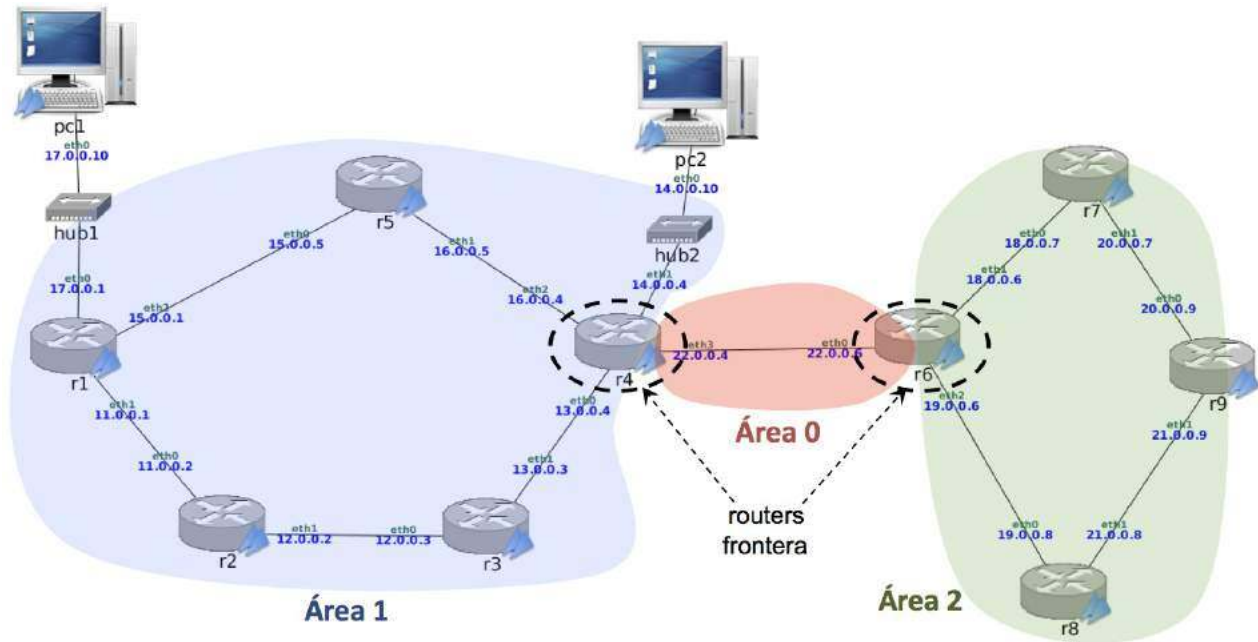
Introducción

- OSPF (*Open Shortest Path First*) es un protocolo de la familia de protocolos del **estado de enlace**.
- OSPF es el protocolo interior recomendado en redes TCP/IP.
- Versión actual: versión 2 (RFC-2328, Abril 1998).
- Los mensajes OSPF se encapsulan directamente dentro de datagramas IP, con número de protocolo 89 (TCP=6, UDP=17)

Jerarquía en OSPF

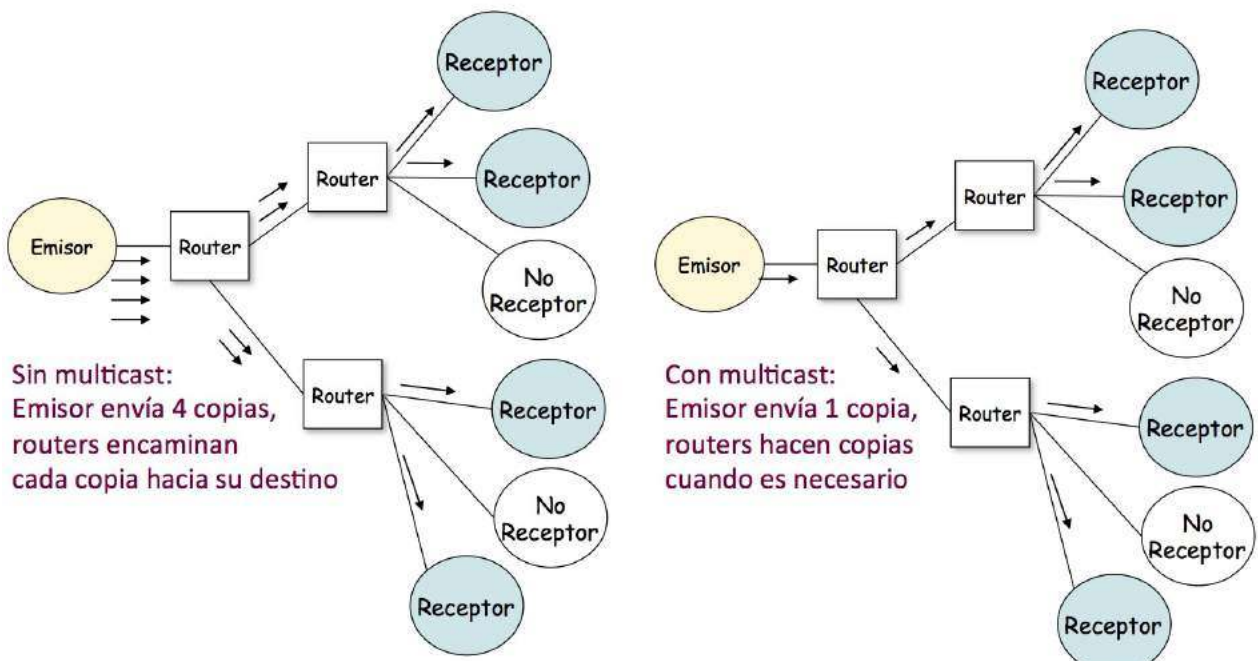
- Puede usarse en sistemas autónomos relativamente grandes.
- Permite el encaminamiento jerárquico definiendo **áreas** dentro del sistema autónomo.

- **ÁREA:** Colección arbitraria de redes, máquinas y *routers*. La topología de un área se mantiene oculta para el resto de áreas. El intercambio de rutas entre áreas se realiza a través del o .
- **BACKBONE o ÁREA 0:** Interconecta todas las demás áreas del sistema autónomo. Cada una de las restantes áreas tendrá un *router* de frontera en el Backbone.



OSPF utiliza IP multicast

Idea de multicast: Un emisor (E) envía un mismo mensaje a un conjunto de receptores (R).



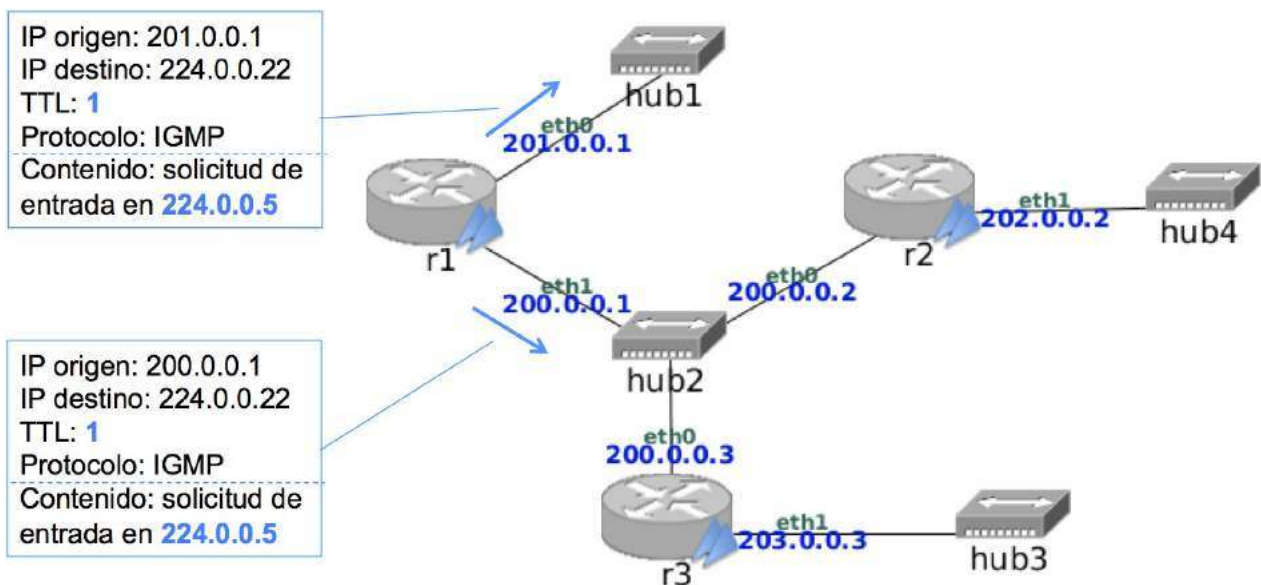
En vez de forzar al emisor a enviar N copias (una por cada receptor) de un determinado mensaje, el emisor envía un único mensaje dirigido a un conjunto de receptores.

El conjunto de receptores se especifica mediante una dirección IP especial, denominada dirección IP de un grupo multicast. El rango de direcciones multicast va desde 224.0.0.0 a 239.255.255.255

Cuando una máquina desea formar parte de un grupo multicast y por tanto recibir los paquetes que van dirigidos a ese grupo, debe utilizar el protocolo IGMP (Internet Group Management Protocol) para enviar su solicitud de pertenencia al grupo. Este mensaje irá dirigido al grupo 224.0.0.22, al que pertenecen todos los *routers* IGMP. Ese mensaje incluye la dirección IP del grupo al que se desea pertenecer.

La dirección IP multicast 224.0.0.5 está reservada para OSPF.

Cuando arranca el *router* OSPF r1 envía (por todas las interfaces donde tiene activado el protocolo OSPF) un mensaje IGMP de solicitud para entrar en el grupo multicast 224.0.0.5



El *router* r1 utilizará la dirección destino 224.0.0.5 para comunicarse con sus *routers* vecinos y enviarles la información de encaminamiento del protocolo OSPF.

De la misma forma, cualquier mensaje de OSPF que un *router* vecino a r1 envíe a la dirección 224.0.0.5 será recibido por r1.

Funcionamiento básico de OSPF

=====

OSPF es un protocolo de encaminamiento interior que mediante el intercambio de mensajes entre los routers de la red permite conocer la topología de dicha red y calcular el camino más corto a todas las subredes para **construir la tabla de encaminamiento** de forma automática.

Los routers OSPF tendrán que ocuparse de las siguientes funciones:

- **Descubrimiento de vecinos** (otros *routers* OSPF conectados a su misma subred) mediante mensajes `HELLO` .
- **Intercambio de la base de datos topológica de OSPF** mediante mensajes `LS UPDATE` (*Link State Update*) que se envían por inundación. Cada *router* con todos los mensajes `LS UPDATE` mantiene una base de datos (*Link State DB*) que representa **topología completa de la red**.
- **Cálculo del algoritmo de Dijkstra en cada router**, partiendo de la base de datos de la topología de la red. Dicho algoritmo permitirá rellenar la tabla de encaminamiento del *router*.
- Los **cambios en la topología** se comunican mediante nuevos mensajes `LS UPDATE` .

Algoritmo de Dijkstra

En una red, dadas las distancias entre cada par de nodos adyacentes, el **Algoritmo de Dijkstra** permite encontrar las rutas de distancia mínima desde cada nodo al resto.

Los protocolos de Estado de Enlace suelen utilizar este algoritmo en cada nodo una vez que dicho nodo ya conoce las distancias entre los nodos adyacentes.

Requiere conocer todas las distancias entre nodos adyacentes.

En OSPF, gracias a los mensajes `LS UPDATE` , cada *router* conocerá la topología completa de la red, y aplicará el algoritmo de Dijkstra para obtener las mejores rutas hacia el resto de nodos. Con el resultado obtenido actualizará su tabla de encaminamiento.

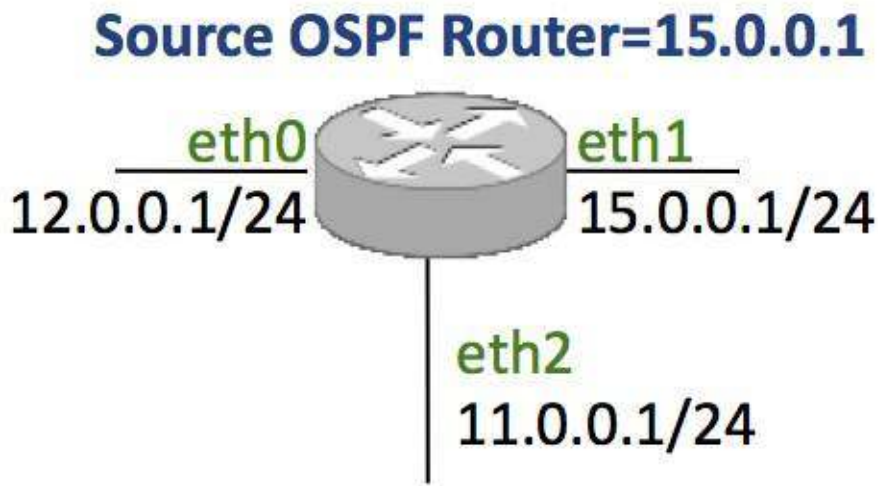
Todos los *routers* partirán de los mismos datos sobre la topología de la red por lo que todas las rutas serán consistentes y óptimas.

Cuando hay cambios en la topología (nuevos enlaces, nuevos *routers*, enlaces que caen, *routers* que se apagan...) los mensajes de estado de enlace harán llegar la información a todos los *routers*, y éstos aplicarán otra vez el algoritmo de Dijkstra para encontrar las nuevas rutas.

Identificador de un *router* OSPF

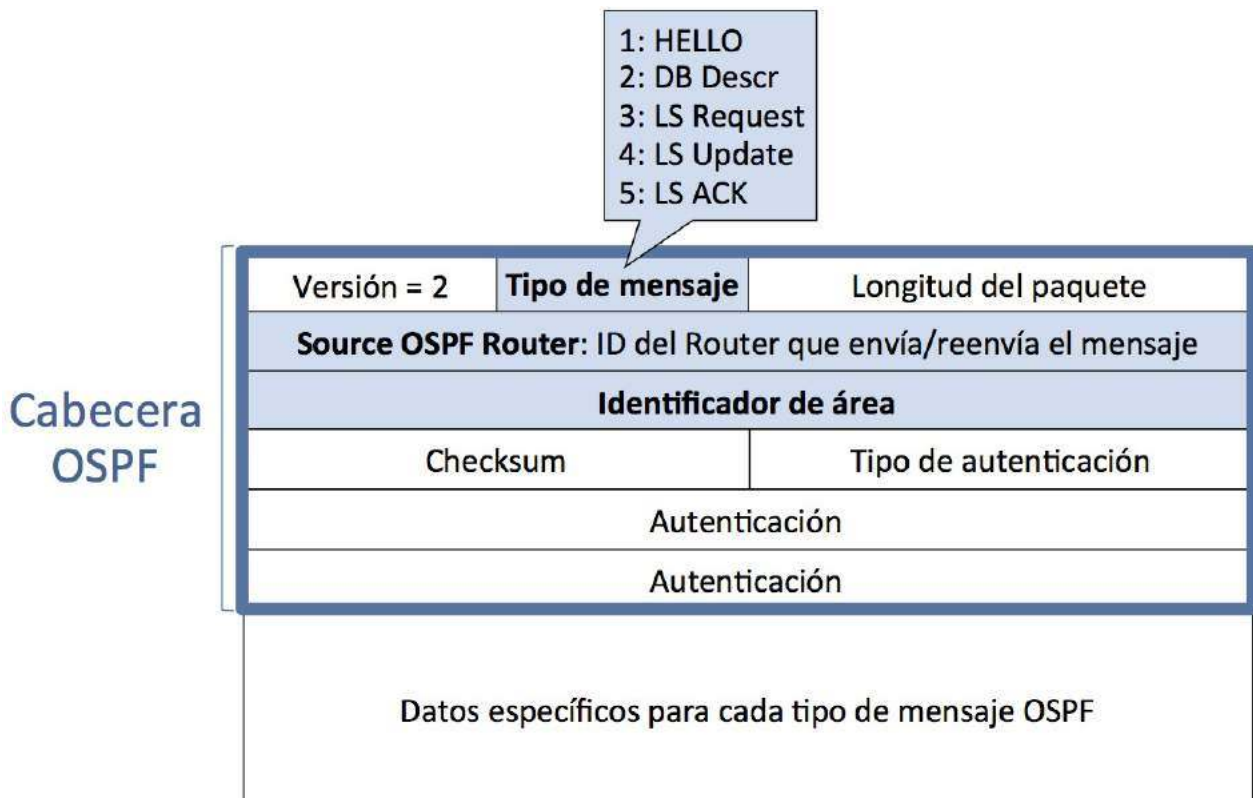
Un *router* OSPF tiene asignado un **identificador de 32 bits**, único en todo su AS. Puede asignarse explícitamente en la configuración del *router*. Es habitual elegir como identificador la dirección IP más alta de las interfaces donde tenga activado OSPF.

Cuando un *router* envía (o reenvía) un mensaje OSPF, escribe su identificador en el campo `Source OSPF Router` de la cabecera de los mensajes OSPF.



Formato de mensaje OSPF

Un mensaje OSPF tendrá la siguiente cabecera obligatoria:



Protocolo HELLO

Los routers OSPF mantienen un protocolo de envío de mensajes `HELLO` con los siguientes objetivos:

- Descubrir nuevos *routers* OSPF vecinos.
- Comprobar que se mantiene la accesibilidad con los *routers* OSPF vecinos ya conocidos.
- Elegir el DR y BDR de cada subred, o informar de cuáles son si ya están elegidos.

Los mensajes `HELLO` se envían por todas las interfaces que tienen activado el protocolo OSPF de un *router*.

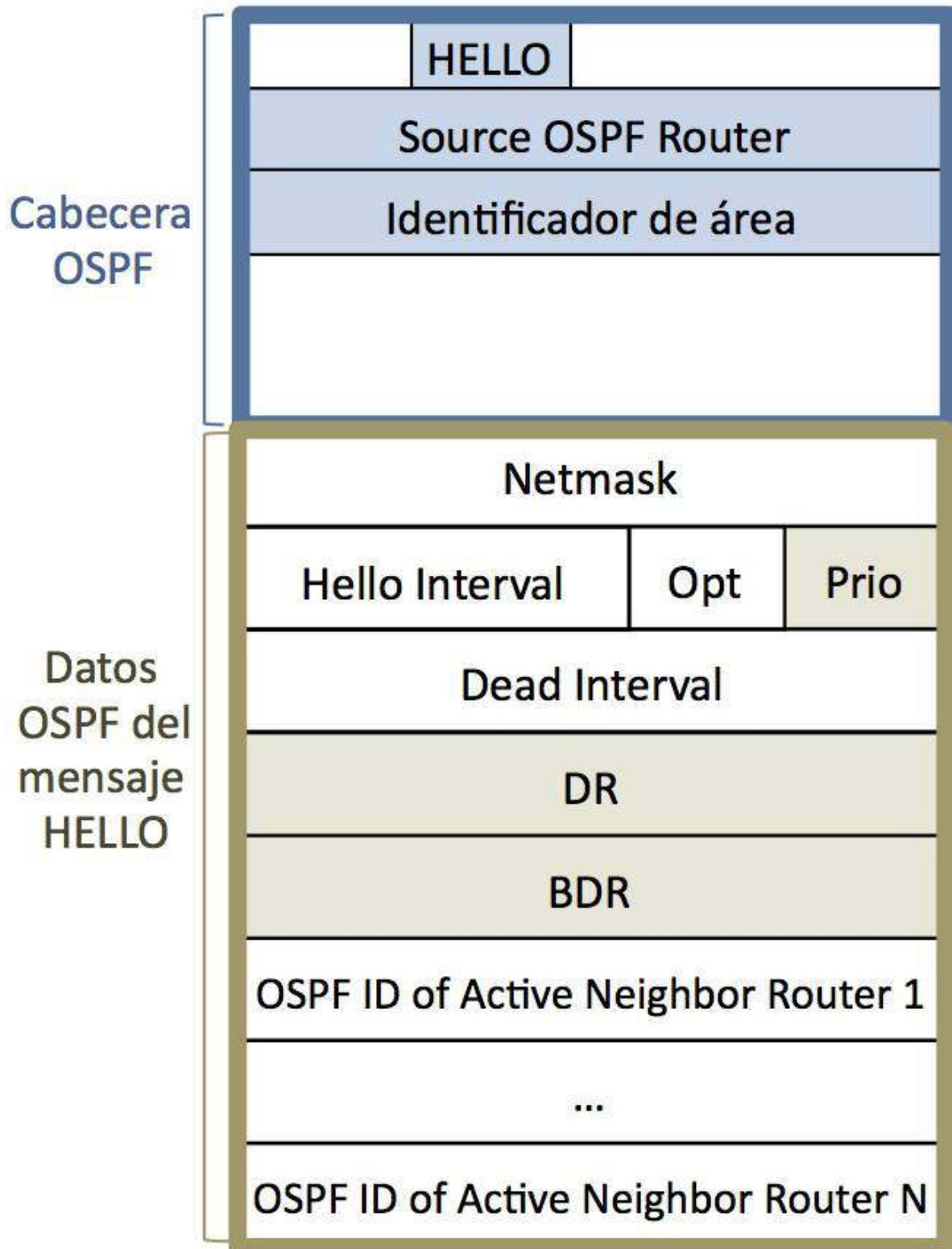
Los mensajes `HELLO` **se envían periódicamente**, cada 10 segundos, dirigidos a la dirección de multicast 224.0.0.5.

Se considera que un vecino está desconectado si no se recibe de él ningún `HELLO` en 40 segundos.

Los mensajes `HELLO` **no se propagan por inundación**, sólo tienen sentido en el enlace local en el que se generan.

Formato de mensaje HELLO

El mensaje HELLO OSPF está formado por la cabecera obligatoria y por los siguientes campo que viajan a continuación:



- Netmask: Máscara de la subred donde se envía el mensaje.
- Hello interval: intervalo en segundos entre mensajes HELLO consecutivos (10 seg)
- Prio: prioridad del *router* que envía el mensaje HELLO para la elección de DR/BDR.

- Dead interval: período en segundos en el que se considera a un vecino OSPF desaparecido si no se recibe de él un nuevo HELLO (40 seg)
- DR: Designated Router
- BDR: Backup Designated Router.
- OSPF id of Active Neighbor i: Identificadores de los routers OSPF vecinos de éste de los que tiene conocimiento (han enviado un HELLO).

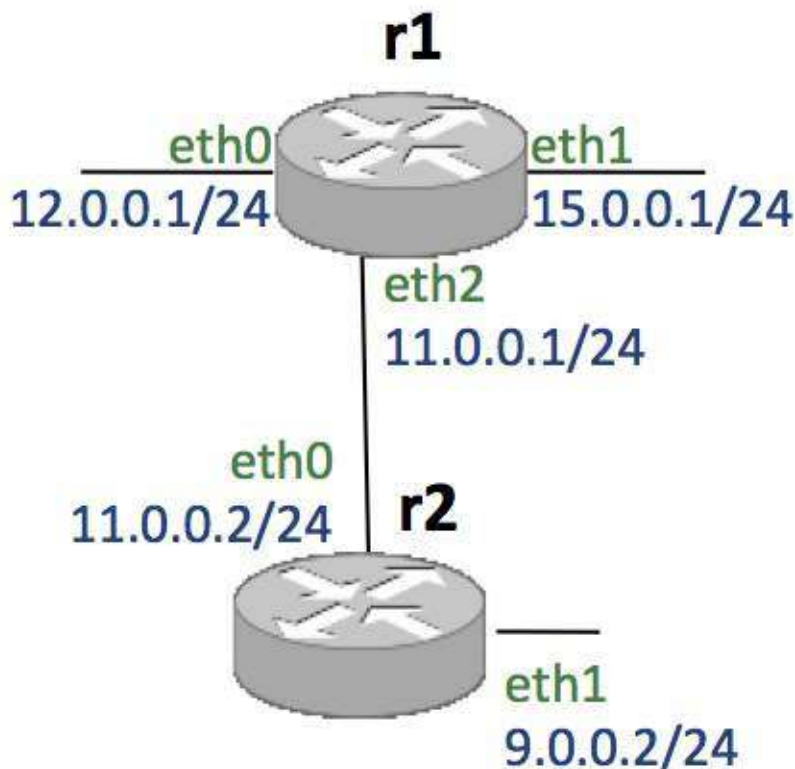
DR (*Designated Router*)

Los mensajes HELLO permiten elegir el DR de la subred por la que se envían.

El **Router Designado (DR, Designated Router) de una subred** es el *router* representante de esa subred y se encarga de crear los mensajes que contienen información sobre ella.

El DR de una subred se expresa con la dirección IP destino dentro de esa subred de uno de los routers que están conectados a ella.

Ejemplo: en la subred 11.0.0.0/24 el DR puede ser 11.0.0.1 o 11.0.0.2.



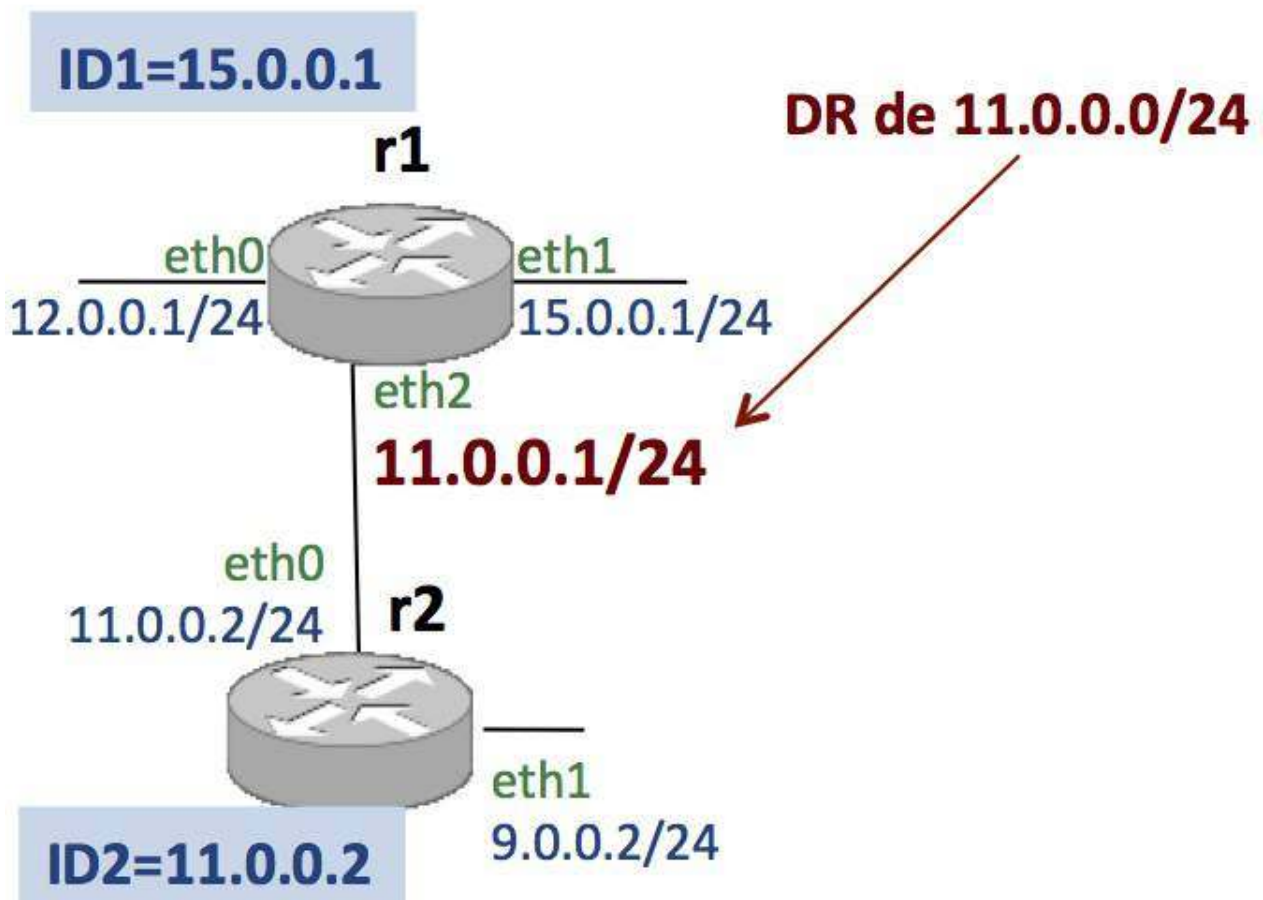
Elección de DR

Si en la red no hay un DR elegido, al arrancar un *router* enviará mensajes `HELLO` con el campo DR vacío (0.0.0.0). Transcurridos 40 segundos se elegirá el DR teniendo en cuenta los siguientes criterios:

Cada *router* elige como DR el *router* que envíe mayor número en el campo `Prio` de los mensajes `HELLO`.

En caso de empate en ese campo, cada *router* elige como DR el que tenga mayor **identificador** (`Source OSPF router`).

Ejemplo: en la subred 11.0.0.0/24 el DR será: 11.0.0.1



Una vez elegido el DR, se coloca su IP en el campo correspondiente de los mensajes `HELLO`.

Si en la red ya hay un DR elegido, al arrancar un *router* éste recibirá mensajes `HELLO` con la dirección IP del DR.

BDR (*Backup Designated Router*)

Los mensajes `HELLO` permiten elegir (adicionalmente al DR) el BDR, que es el DR “de reserva”.

Se elige como BDR el segundo mejor *router* según los criterios de elección de DR.

Una vez elegido BDR, la dirección IP del BDR en esa subred se enviará en el campo BDR de los mensajes de HELLO .

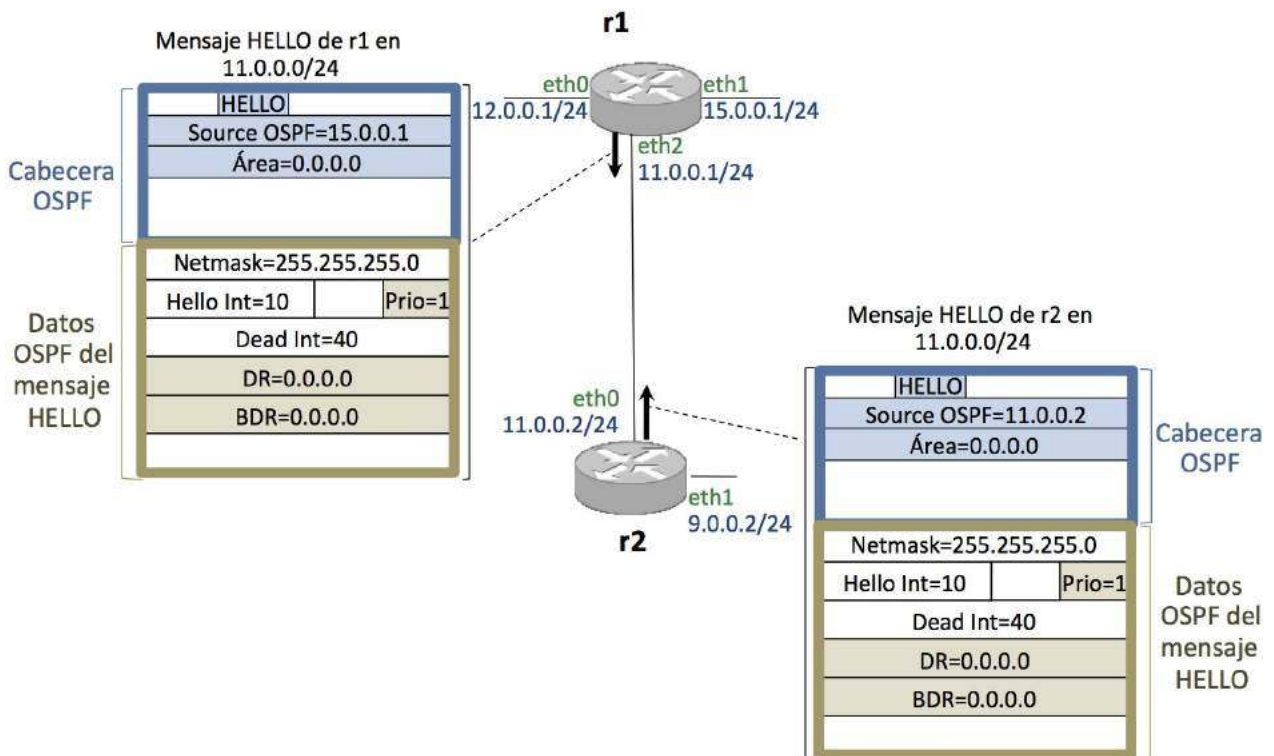
Si el DR deja de funcionar (deja de enviar un HELLO en 40 segundos), el BDR se convierte en el nuevo DR y se elegirá un nuevo BDR.

Una vez elegidos DR y BDR en una subred si se conecta un nuevo router a esa subred, no se modifica el DR ni el BDR, incluso aunque los *routers* que se conecten tengan mayor prioridad o mayor identificador.

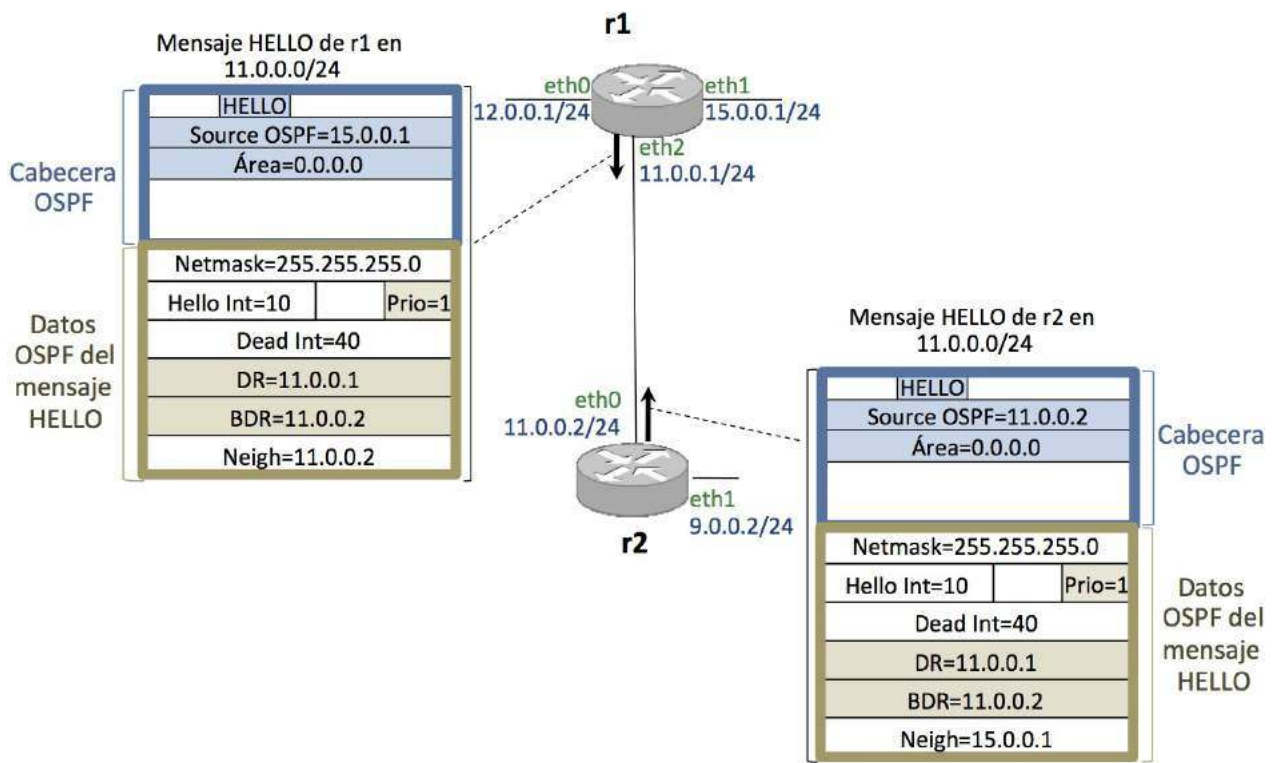
Si en una subred sólo hay conectado un *router* OSPF, éste se elegirá como DR y no habrá BDR. Si posteriormente arrancan otros *routers* OSPF conectados a esa subred, se elegirá entre ellos el BDR.

Ejemplo: elección de DR y BDR

r1 y r2 comienzan a ejecutar OSPF simultáneamente. Inicialmente no hay elegidos ni DR ni BDR y ambos routers intercambian mensajes de HELLO para darse a conocer.

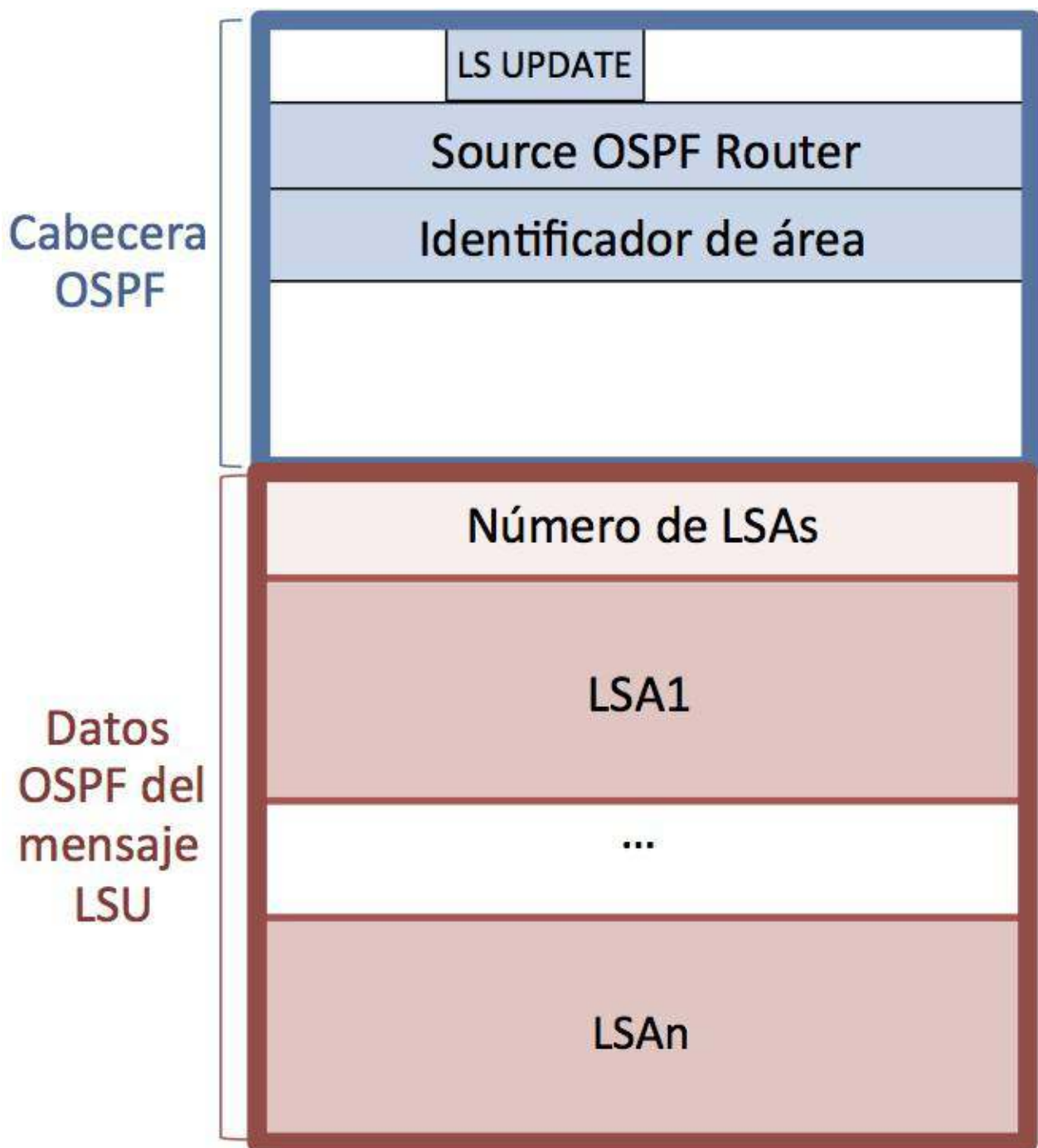


Transcurridos aproximadamente 40 segundos, ambos saben que hay otro router en la misma subred y proceden a elegir DR y BDR, en este caso como los routers tienen la misma prioridad, se elegirá DR/BDR según su identificador. El mayor identificador es el de r1 que se convierte en DR. r2 se convertirá en BDR.



Mensajes LS UPDATE (*Link State Update*)

Un mensaje `LS UPDATE` contiene información de estado de enlace que permite a los *routers* OSPF reconstruir la topología de la red del AS. La información que contiene un `LS UPDATE` está representada por 1 o más LSA (*Link State Advertisements*), Anuncios de Estado de Enlace.



Estudiaremos distintos tipos de LSA:

- **Router LSA (Router Link State Advertisement):** Cada *router* OSPF genera un LSA de este tipo para informar de las interfaces que tiene configuradas.
- **Network LSA (Network Link State Advertisement):** El DR de cada subred que contenga dos o más *routers* OSPF genera un LSA de este tipo para informar de los *routers* que se encuentran conectados a dicha subred.

Los LSA se envían siempre contenidos dentro de un mensaje `LS UPDATE`.

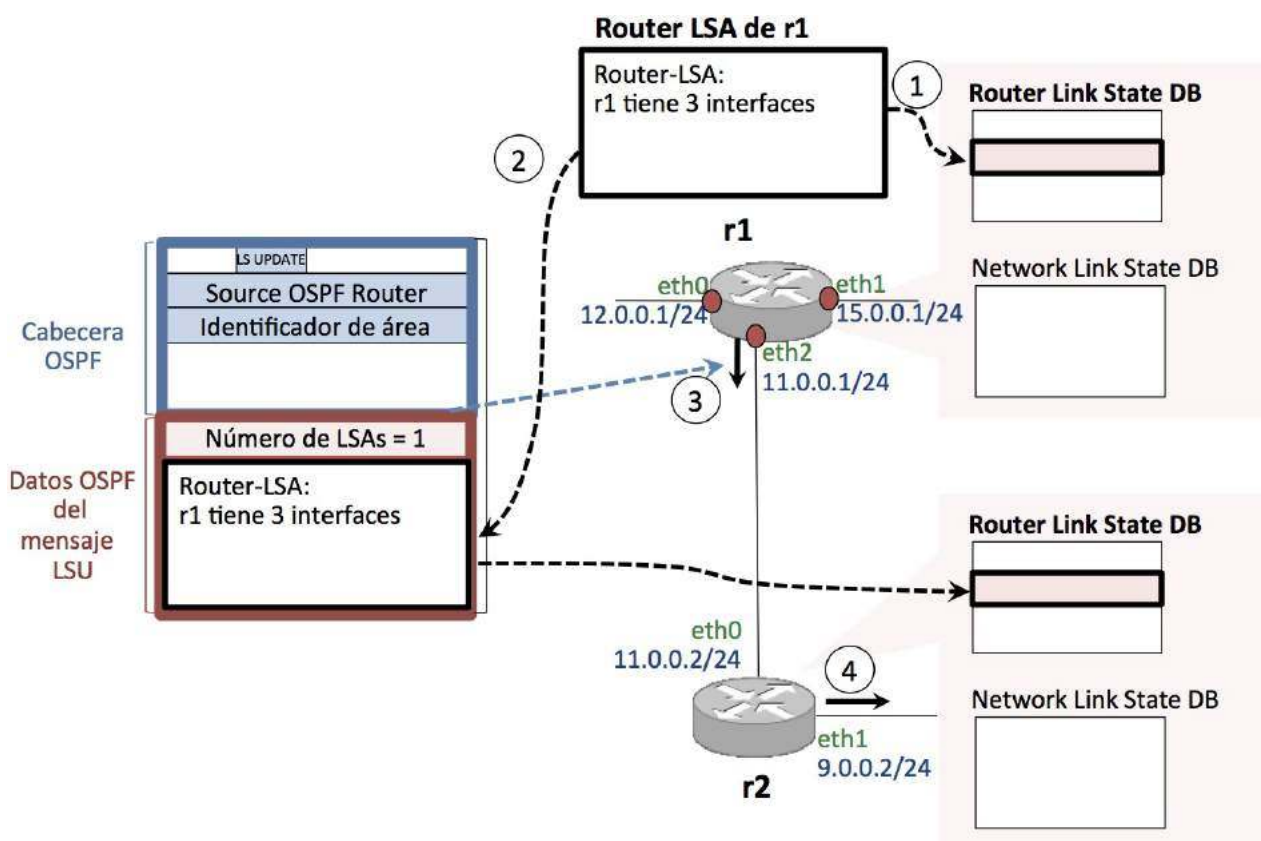
Cada *router* almacena todos los LSA (los suyos y los que recibe de otros *routers*) en una base de datos, una por cada tipo de LSA

- Router Link State DB : para los mensajes Router LSA
- Network Link State DB : para los mensajes Network LSA

Router LSA

Cuando un *router* genera un Router LSA:

1. Lo almacena en su propia base de datos Router Link State Database
2. Genera un mensaje LS UPDATE con el LSA
3. Lo envía SÓLO por las interfaces donde sabe que hay otros *routers* OSPF vecinos.
4. Es un envío **: los que lo reciban lo reenviarán a sus vecinos (y lo almacenarán en su DB), y así sucesivamente

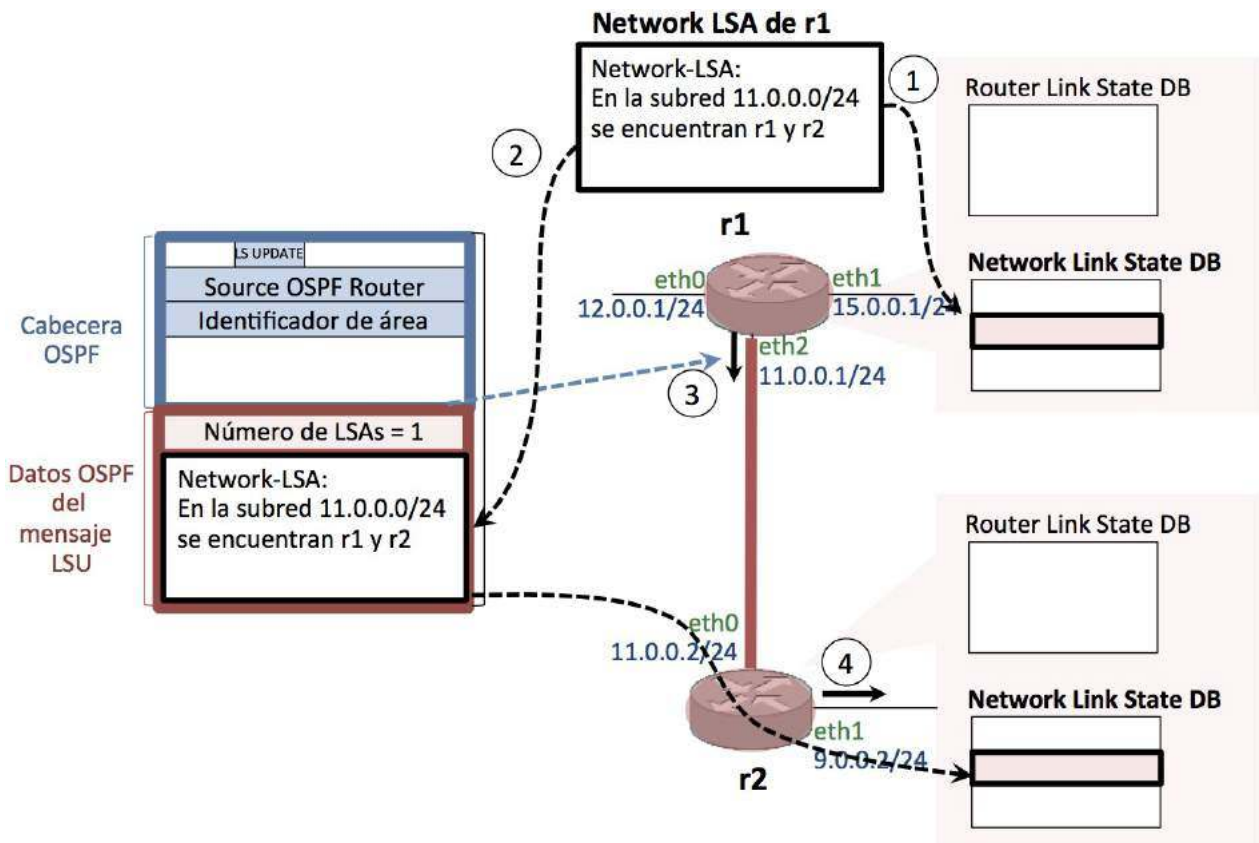


Network LSA

Cuando el *router* DR de una subred genera un Network LSA:

1. Lo almacena en su base de datos Network Link State Database
2. Genera un mensaje LS UPDATE con el LSA
3. Lo envía SÓLO por las interfaces donde sabe que hay otros *routers* OSPF vecinos.

4. Es un envío **: los que lo reciban lo reenviarán a sus vecinos (y lo almacenarán en su DB), y así sucesivamente.



Número de secuencia de un LSA

Cada LSA queda identificado por estos 3 campos:

- **LS Type** : Tipo de LSA
- **Link State ID** : Info. dependiente del tipo de LSA
- **Advertising router** : Identificador del router que lo ha creado

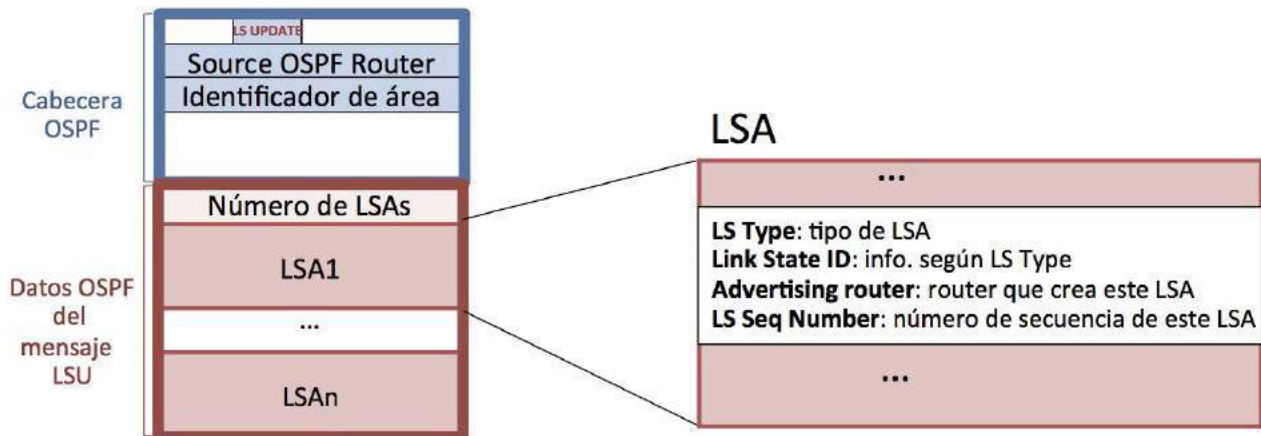
El *router* que genera un LSA le asigna un número de secuencia, que viajará en el propio LSA.

Ningún otro *router* que reciba el LSA modificará el valor del número de secuencia durante el proceso de envío por inundación.

Los espacios de números de secuencia que un *router* utiliza para cada LSA son diferentes

Cuando cambie la información que contiene un LSA (por cambiar la configuración de la red), el router lo enviará de nuevo, con la nueva información, y con un número de secuencia una unidad mayor que el antiguo.

El número de secuencia se utiliza para saber si un mensaje es antiguo:: Si dos mensajes son del mismo tipo y han sido generados por un mismo *router*, el mensaje cuyo número de secuencia sea mayor será el mensaje más moderno.



Encaminamiento por inundación

En general, el **envío por inundación** se utiliza cuando aún no se dispone de otra información para encaminar (ejemplo: al arrancar o como paso intermedio de algún protocolo de encaminamiento).

Funcionamiento básico:

1. Cada paquete recibido por un nodo es reenviado a todos los vecinos excepto al que se lo envió a él.
2. Los paquetes van etiquetados y numerados.
3. Si un nodo recibe un paquete que ya ha reenviado, lo descarta.

Inundación de LSAs en OSPF

Todos los mensajes OSPF llevan TTL=1. Por tanto, para que la inundación se realice, cuando un *router* recibe un LSA:

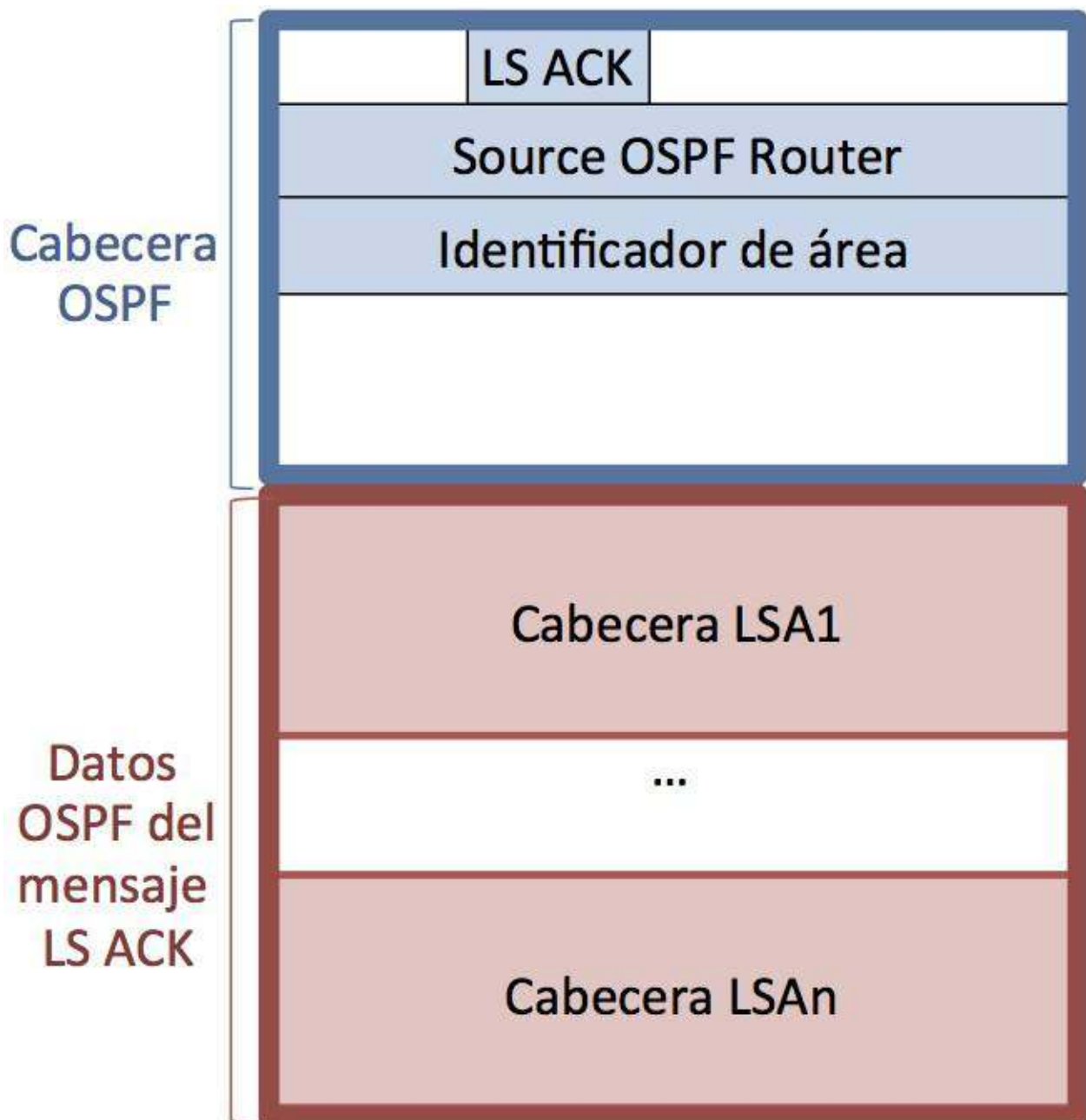
- si ya lo tenía en su base de datos, o es más antiguo que el que tiene, lo descartará y no lo reenviará
- si no lo tenía en su base de datos, o es más nuevo que el que tiene, lo almacenará (sustituyendo el mensaje antiguo en su caso) y lo reenviará **SÓLO** por las interfaces donde hay otros *routers* OSPF vecinos. No lo reenviará por la interfaz por donde lo había recibido.

Para comparar LSAs se utilizan los 3 campos que lo identifican (`LS Type` , `Link State ID` , `Advertising Router`) junto con el número de secuencia:

- **Un LSA recibido es obsoleto** si su número de secuencia es **menor o igual** que el último número de secuencia almacenado en la BD para ese mismo LSA (mismos 3 campos que lo identifican).
 - **Un LSA recibido es nuevo** si su número de secuencia es **mayor** que el último número de secuencia almacenado en la BD para ese mismo LSA (mismos 3 campos que lo identifican), o bien si un LSA con esos 3 campos **aún no estaba** en la BD.

Fiabilidad en la inundación de LSAs

Para proporcionar fiabilidad a la transmisión de mensajes LSA Update se utilizan acks, mensajes LSA ACKs.



- Cada LSA contenido en un mensaje `LS UPDATE` debe ser asentido con un mensaje `LS ACK`, enviado a la dirección 224.0.0.5. Un `LS ACK` puede asentir varios LSA.
- No es necesario que todos los LSA contenidos en un único mensaje `LS UPDATE` sean asentidos con un único mensaje `LS ACK`.
- Si no se recibe de algún *router* un `LS ACK` para un cierto LSA en 5 segundos, se reenviará dicho LSA en un nuevo `LS UPDATE` (los reenvíos se realizan de forma unicast al *router* que no ha asentido el LSA).

Bases de datos de OSPF

- **Router Link States DB:**

En esta base de datos hay **una entrada por cada router OSPF** de la red, indicando los datos de cada una de sus interfaces. Cada entrada contiene el **último mensaje**

Router-LSA enviado por cada *router* OSPF.

|Router Link States DB|

|Último Router-LSA de Router 1| |Último Router-LSA de Router 2| |...| |Último Router-LSA de Router n|

- **Network Link States DB:**

En esta base de datos hay **una entrada por cada subred en la que hay más de un router OSPF**, indicando los *routers* OSPF que están conectados en dicha subred.

Cada entrada contiene el **último mensaje** **Network-LSA** enviado por el *router* DR de cada una de las subredes en la que hay más de un *router* OSPF conectado.

|Network Link States DB|

|Último Network-LSA del DR de la subred 1| |Último Network-LSA del DR de la subred 2| |...| |Último Network-LSA del DR de la subred m|

Campos de un *Router LSA*

Un **Router LSA** contiene la siguiente información (tal y como la muestra `wireshark`):

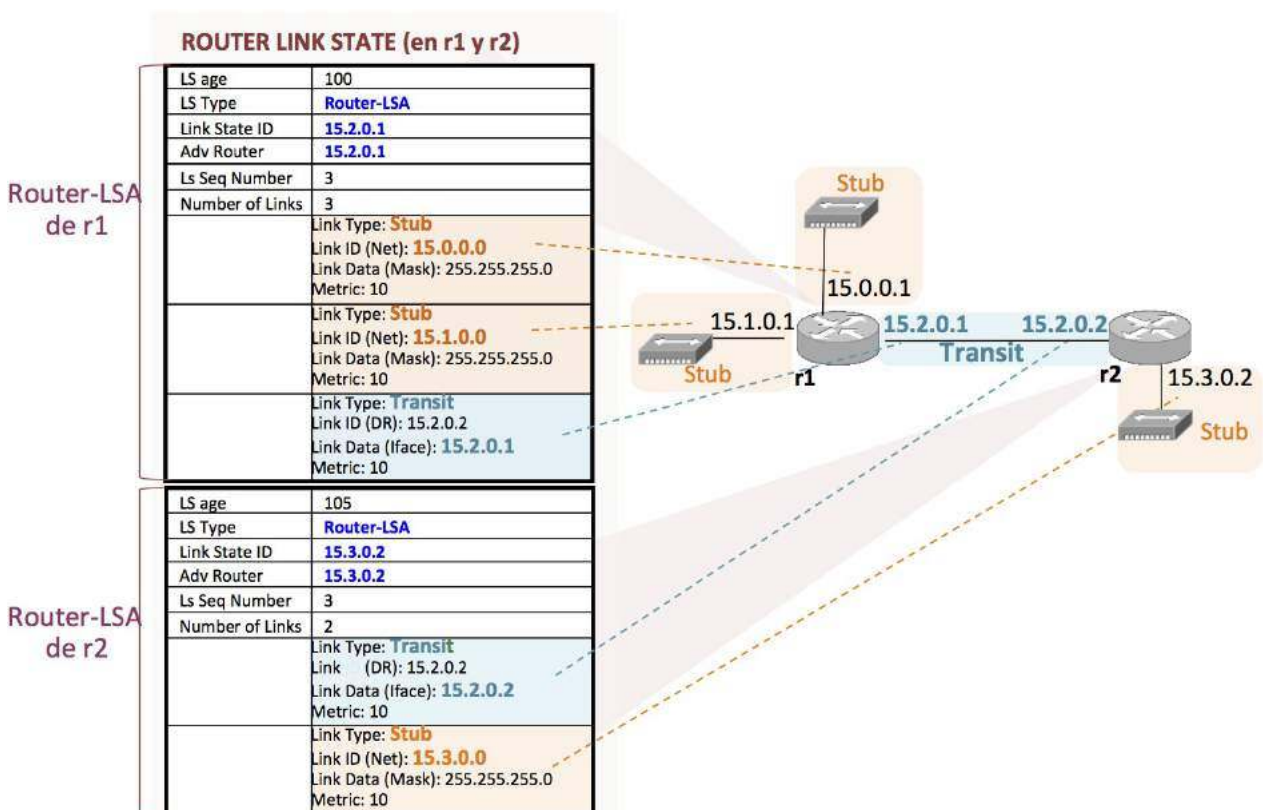
LS age	número de segundos que han pasado desde que el LSA fue generado. Este valor aumenta: cada vez que un <i>router</i> reenvía (inundación) un anuncio generado por otro <i>router</i> (aumenta un segundo) y cuando se almacena en una base de datos de un <i>router</i> (aumenta según van pasando los segundos)
LS Type	router-LSA
Link State ID	ID del <i>router</i> que generó el anuncio
Advertising router	ID del <i>router</i> que generó el anuncio
LS Seq Number	número de secuencia
Number of Links	número de interfaces del <i>router</i>
Link Type (1)	Dos tipos: Stub: No hay otros <i>routers</i> OSPF en esa interfaz Transit: Hay otros <i>routers</i> OSPF en a esa interfaz
Link ID (1)	En Stub: Red a la que está conectado el <i>router</i> . En Transit: DR de esa subred (su IP)
Link Data (1)	En Stub: Máscara En Transit: IP de este <i>router</i> en esa subred
Metric (1)	Coste (10 por defecto)
Link Type (2)	...
...	...
Link Type (n)	...
...	...

Base de datos: *Router Link States*

Cada *router* tiene una base de datos con la información de las interfaces de todos los *routers* OSPF.

Router-LSA Router 1	LS age	número de segundos que han pasado desde que el LSA fue generado. Este valor aumenta: cada vez que un <i>router</i> reenvía (inundación) un anuncio generado por otro <i>router</i> (aumenta un segundo) y cuando se almacena en una base de datos de un <i>router</i> (aumenta según van pasando los segundos)
	LS Type	router-LSA
	Link State ID	ID del <i>router</i> que generó el anuncio
	Advertising router	ID del <i>router</i> que generó el anuncio
	LS Seq Number	número de secuencia
	Number of Links	número de interfaces del <i>router</i>
	Link Type (1)	Dos tipos: Stub: No hay otros <i>routers</i> OSPF en esa interfaz Transit: Hay otros <i>routers</i> OSPF en a esa interfaz
	Link ID (1)	En Stub: Red a la que está conectado el <i>router</i> . En Transit: DR de esa subred (su IP)
	Link Data (1)	En Stub: Máscara En Transit: IP de este <i>router</i> en esa subred
	Metric (1)	Coste (10 por defecto)
Link Type (2)	...	
...	...	
Link Type (n)	...	
...	...	
Router-LSA Router 2		...
...		...
Router-LSA Router n		...

Base de datos: *Router Link States*



Campos de un *Network LSA*

Un **Router LSA** contiene la siguiente información (tal y como la muestra `wireshark`):

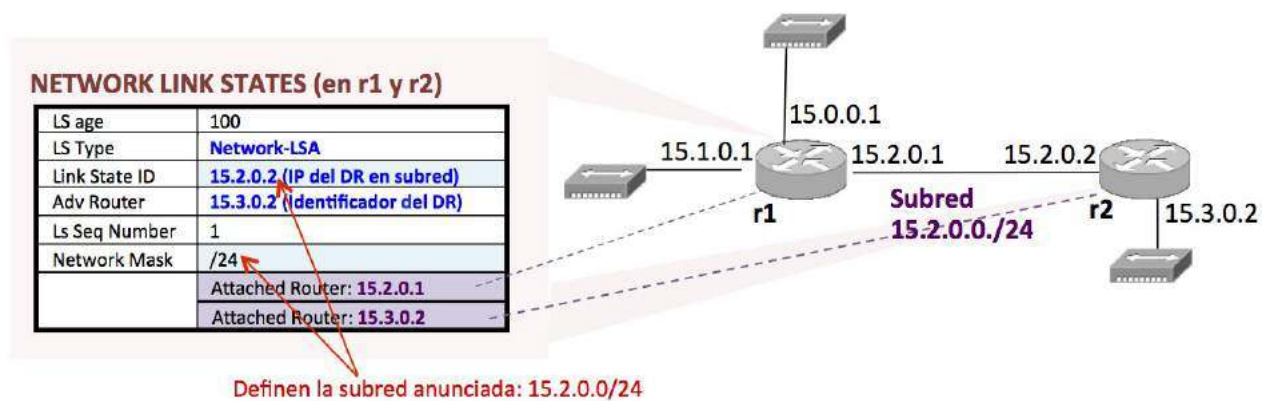
LS age	número de segundos que han pasado desde que el LSA fue generado. Este valor aumenta: cada vez que un <i>router</i> reenvía (inundación) un anuncio generado por otro <i>router</i> (aumenta un segundo) y cuando se almacena en una base de datos de un <i>router</i> (aumenta según van pasando los segundos)
LS Type	<code>network-LSA</code>
Link State ID	DR que generó el anuncio (su IP)
Advertising router	ID del <i>router</i> que generó el anuncio (ID del DR)
LS Seq Number	número de secuencia
Network Mask	máscara de la subred
	Attached Router: ID del <i>router</i> conectado a esa subred
	Attached Router: ...
	Attached Router: ...

Base de datos: *Network Link States*

Cada *router* tiene una base de datos con la información de las subredes en las que hay más de un *router* OSPF, indicando qué *routers* se encuentran conectados en cada una de esas subredes.

Network-LSA 1	LS age	número de segundos que han pasado desde que el LSA fue generado. Este valor aumenta: cada vez que un <i>router</i> reenvía (inundación) un anuncio generado por otro <i>router</i> (aumenta un segundo) y cuando se almacena en una base de datos de un <i>router</i> (aumenta según van pasando los segundos)
	LS Type	<code>network-LSA</code>
	Link State ID	IP en la red que se anuncia del <i>router</i> que generó el anuncio
	Advertising router	ID del <i>router</i> que generó el anuncio (ID del DR)
	LS Seq Number	número de secuencia
	Network Mask	máscara de la subred
		Attached Router: ID del <i>router</i> conectado a esa subred
		Attached Router: ...
Network-LSA 2		...
...		...
Network-LSA 3		...

Base de datos: *Network Link States*



Caducidad de los mensajes LSU

Cuando se crea un LSA, su campo `LS Age` se pone a 0. Este campo representa el número de segundos que ha pasado desde la creación del LSA.

Cada vez que se reenvía un LSA entre *routers*, su `LSA Age` aumenta en 1.

Cuando un LSA está almacenado en una BD, su `LSA Age` aumenta en 1 por cada segundo que pase.

Un LSA caduca cuando su `LS Age` llega a 3600 (una hora), y en ese momento debe eliminarse la base de datos, recalculándose de nuevo Dijkstra.

Los *routers* OSPF deben refrescar cada 1800 segundos (media hora) los LSA que han creado, reenviándolos en nuevos mensajes `LS UPDATE`.

Intercambio inicial de las bases de datos de OSPF

Cuando dos *routers* OSPF vecinos se ven por primera vez a través de los mensajes `HELLO`, comienzan a intercambiarse el contenido de sus respectivas bases de datos.

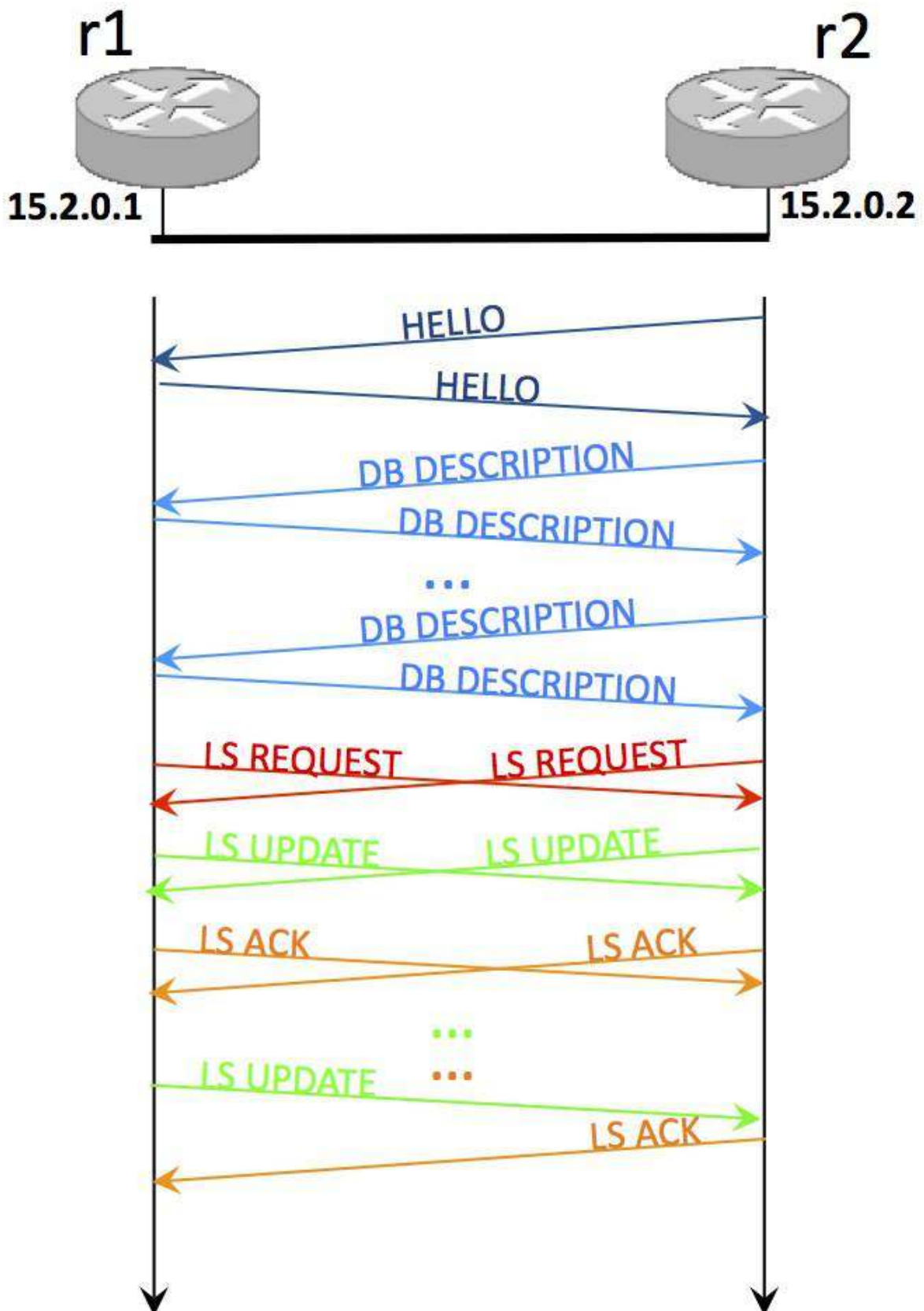
Este intercambio se realiza a través de mensajes `DB Description` enviados de forma unicast.

Puede llegar a ser un proceso complejo y no entraremos en los detalles. De forma general:

- Cada *router* especifica la lista de mensajes LSA que hay en sus bases de datos, indicando no el contenido sino sólo los 3 campos que identifican cada LSA.
- Cada *router* compara la lista de identificadores de LSA que recibe con los que tiene almacenados en sus bases de datos y pide mediante mensajes `LS REQUEST` al otro *router* los LSA que le faltan.

- Cada *router* envía los LSA pedidos mediante mensajes `LS UPDATE`.

Intercambio inicial de las bases de datos de OSPF: ejemplo



`r1` lleva arrancado un tiempo. Al arrancar `r2` envía un mensaje `HELLO` .

Cuando `r1` y `r2` descubren que son vecinos, se intercambian la lista de LSAs que tienen en sus bases de datos a través de los mensajes `DB DESCRIPTION` (unicast).

Cada uno le pide al otro los LSA que le faltan mediante mensajes `LS REQUEST` (unicast).

Cada uno envía los LSAs pedidos en mensajes `LS UPDATE` (multicast).

Los LSAs recibidos en `LS UPDATE` hay que asentirlos con mensajes `LS ACK` (multicast).

Modificaciones en las bases de datos de OSPF

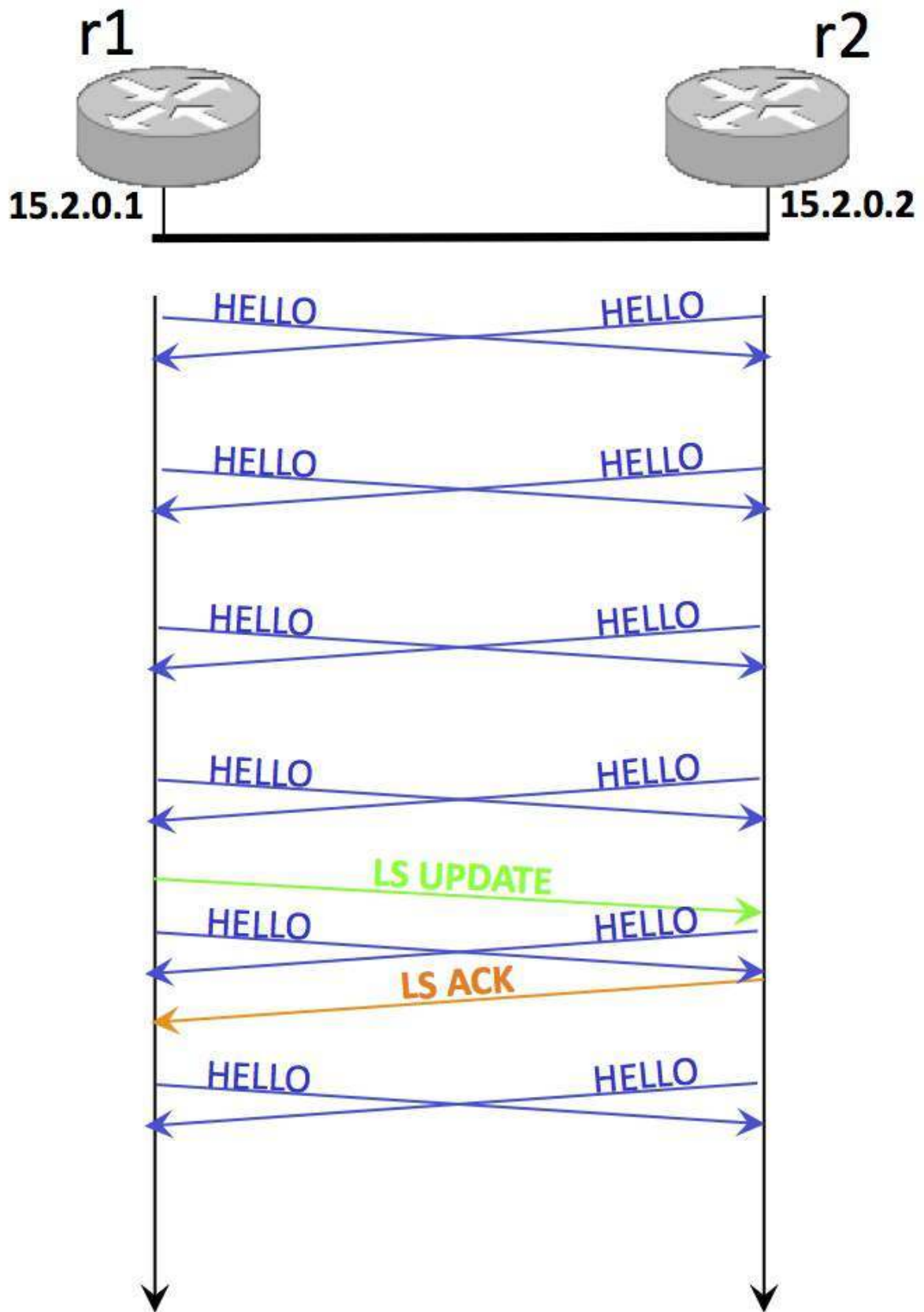
Mientras no haya cambios en la topología, cada *routers* OSPF sólo envía:

- cada 10 segundos mensajes `HELLO` por todas sus interfaces OSPF
- cada 1800 segundos mensajes `LS UPDATE` refrescando los LSA que ha creado

Cuando se produce algún cambio en la topología de la red (se arranca/apaga un *router* OSPF, una interfaz queda inaccesible...), este cambio se propaga a través de mensajes `LS UPDATE` que incluyen los LSAs modificados.

Cada vez que un *router* recibe nuevos LSAs debe recalcular el algoritmo de Dijkstra para actualizar su tabla de encaminamiento. Como es un proceso pesado, hay un tiempo mínimo que tiene que pasar entre aplicaciones sucesivas del algoritmo.

Modificaciones en las bases de datos de OSPF: ejemplo



r1 y r2 llevan arrancados un tiempo.

Cuando r1 detecta un cambio en la topología, envía un LS UPDATE con los LSAs modificados.

El *router* r2 asentirá los LSAs a través de un mensaje LS ACK .

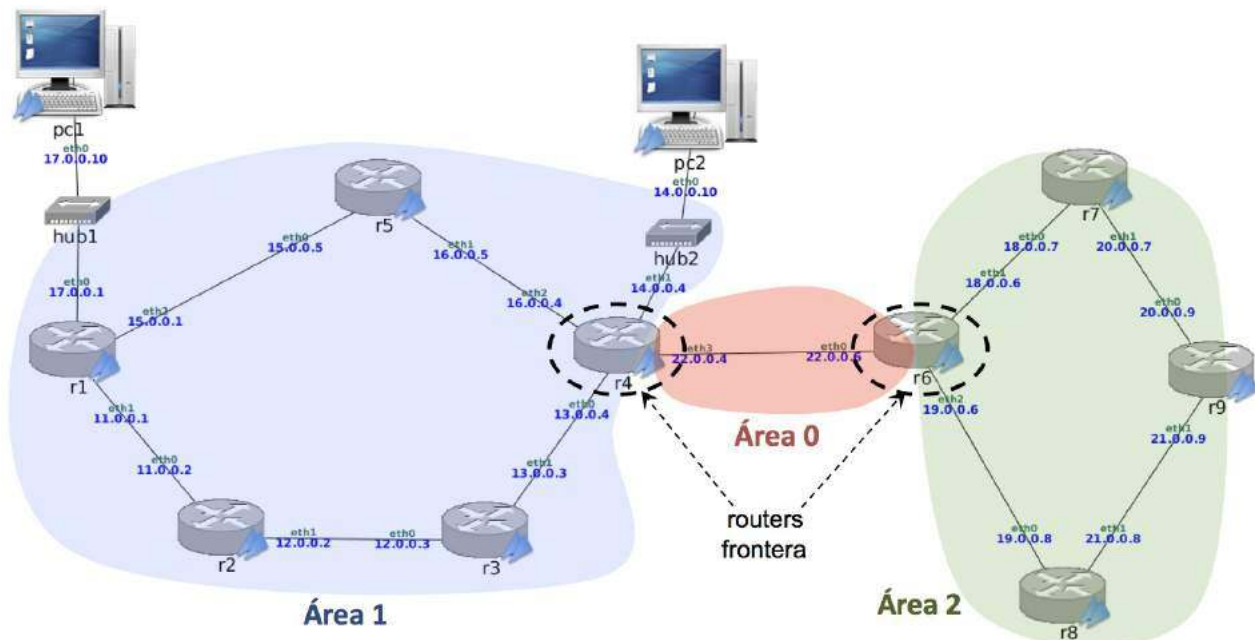
Mensajes entre diferentes áreas OSPF

Áreas OSPF

En OSPF los *routers* se pueden agrupar en áreas, donde un área queda definida por el conjunto de *routers* que comparten el mismo identificador de área.

El área 0 corresponde al *backbone* al que se conectan todas las áreas.

Los ABR (*Area Border Router*), *routers* frontera de un área, tendrán al menos una interfaz de red conectada al área 0 y otra interfaz conectada a otra área diferente.



Los LSA que describen la topología de un área (Router LSA , Network LSA) describen sólo las interfaces y redes de un área, y se transmiten en mensajes LS UPDATE únicamente dentro de ese área.

Entre áreas diferentes la información se transmite con un **nuevo tipo de LSA: Summary LSA**. Estos LSA informan dentro de un área de las subredes que existen fuera de esa área (en otras áreas).

- Los `summary LSA` son generados por los ABR.
- Los `summary LSA` no contiene tanta información como los `Router LSA` y los `Network-LSA` , por lo que dentro de un área cada *router* podrá reconstruir la topología de esa área, pero no del resto de áreas. De las redes de fuera de su área solo tendrá una

información “tipo protocolo de vector de distancias”: sabe que existe la red, su métrica, y que para llegar a ella debe ir a través del ABR que generó el `Summary LSA` .

Summary-LSA

LS age	número de segundos que han pasado desde que el LSA fue generado. Este valor aumenta: cada vez que un <i>router</i> reenvía (inundación) un anuncio generado por otro <i>router</i> (aumenta un segundo) y cuando se almacena en una base de datos de un <i>router</i> (aumenta según van pasando los segundos)
LS Type	<code>summary-LSA</code>
Link State ID	Dirección de la subred que se anuncia
Advertising router	ID del <i>router</i> que generó el anuncio
LS Seq Number	número de secuencia
Netmask	Máscara de la subred que se anuncia
Metric	Métrica para esa subred

Base de datos: Summary-LSA

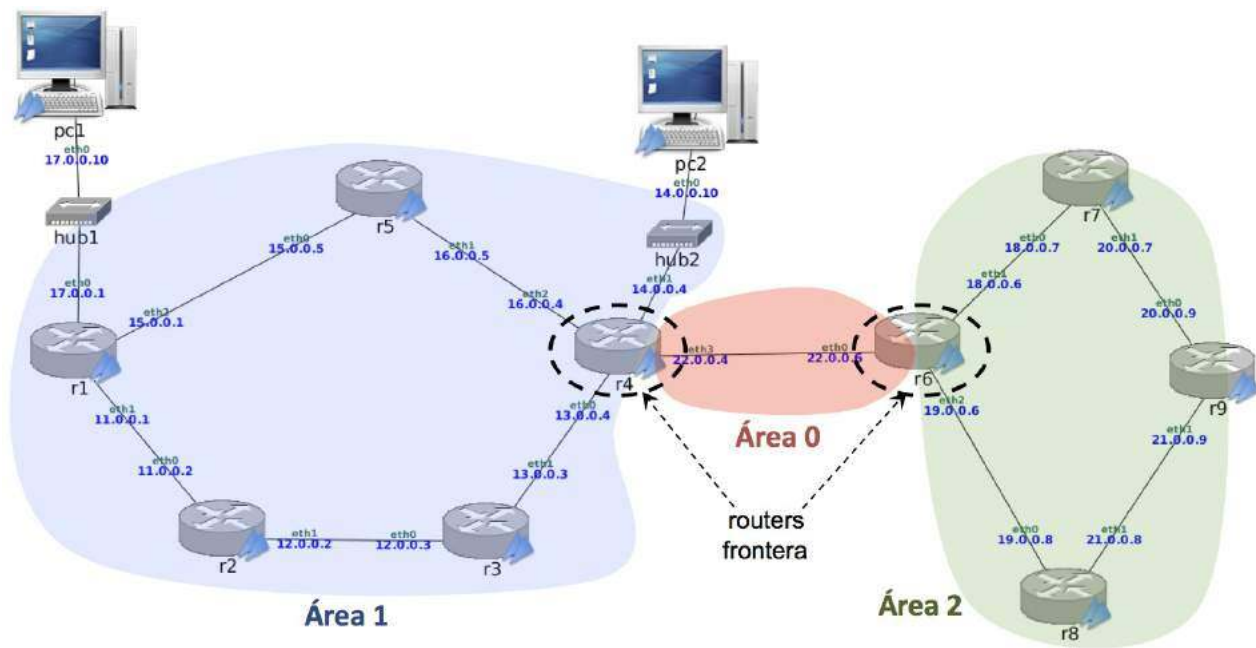
Cada *router* tiene una base de datos con la información de los mensajes Summary-LSA, donde se informa de las subredes que existen en el resto de las áreas.

Las bases de datos de Summary-LSA son diferentes en *routers* que pertenecen a distinta área. Es la misma para los *routers* del mismo área.

Summary-LSA 1	LS age	número de segundos que han pasado desde que el LSA fue generado. Este valor aumenta: cada vez que un <i>router</i> reenvía (inundación) un anuncio generado por otro <i>router</i> (aumenta un segundo) y cuando se almacena en una base de datos de un <i>router</i> (aumenta según van pasando los segundos)
	LS Type	<code>summary-LSA</code>
	Link State ID	Dirección de la subred que se anuncia
	Advertising router	ID del <i>router</i> que generó el anuncio
	LS Seq Number	número de secuencia
	Netmask	Máscara de la subred que se anuncia
	Metric	Métrica para esa subred
Summary-LSA 2	...	
...	...	
Summary-LSA 3	...	

Ejemplos de LSAs cuando hay varias áreas

En la siguiente figura `r4` y `r6` son ABR y serán los responsables de generar Summary-LSA.



Área 0

En el área 0 se habrán generado y propagado dentro del área 0 los siguientes LSAs:

- Router-LSA de r4 : describe la interfaz r4-eth3 .
- Router-LSA de r6 : describe la interfaz r6-eth0 .
- Network-LSA de la red 22.0.0.0/24 (creado por r4 , DR)

En el área 0 se habrán recibido y propagado dentro del área los siguientes LSAs, estos LSAs se corresponden con información de subredes de otras áreas diferentes al área 0:

- Summary-LSA de la red 11.0.0.0/24 (creado por r4) .
- Summary-LSA de la red 12.0.0.0/24 (creado por r4)
- Summary-LSA de la red 13.0.0.0/24 (creado por r4)
- Summary-LSA de la red 14.0.0.0/24 (creado por r4)
- Summary-LSA de la red 15.0.0.0/24 (creado por r4)
- Summary-LSA de la red 16.0.0.0/24 (creado por r4)
- Summary-LSA de la red 17.0.0.0/24 (creado por r4)
- Summary-LSA de la red 18.0.0.0/24 (creado por r6)
- Summary-LSA de la red 19.0.0.0/24 (creado por r6)
- Summary-LSA de la red 20.0.0.0/24 (creado por r6)
- Summary-LSA de la red 21.0.0.0/24 (creado por r6)

Área 1

En el área 1 se habrán generado y propagado dentro del área 1 los siguientes LSAs:

- Router-LSA de r1 , describiendo todas sus interfaces.
- Router-LSA de r2 , describiendo todas sus interfaces.
- Router-LSA de r3 , describiendo todas sus interfaces.
- Router-LSA de r4 , describiendo sus interfaces r4-eth0, r4-eth1, r4-eth2 .
- Router-LSA de r5 , describiendo todas sus interfaces.
- Network-LSA de la red 11.0.0.0/24 (creado por r1 , DR)
- Network-LSA de la red 12.0.0.0/24 (creado por r3 , DR)
- Network-LSA de la red 13.0.0.0/24 (creado por r4 , DR)
- Network-LSA de la red 15.0.0.0/24 (creado por r1 , DR)
- Network-LSA de la red 16.0.0.0/24 (creado por r4 , DR)

En el área 1 se habrán recibido y propagado dentro del área los siguientes LSAs, estos LSAs se corresponden con información de subredes de otras áreas diferentes al área 1:

- Summary-LSA de la red 18.0.0.0/24 (creado por r4)
- Summary-LSA de la red 19.0.0.0/24 (creado por r4)
- Summary-LSA de la red 20.0.0.0/24 (creado por r4)
- Summary-LSA de la red 21.0.0.0/24 (creado por r4)
 - Summary-LSA de la red 22.0.0.0/24 (creado por r4)

Área 2

En el área 2 se habrán generado y propagado dentro del área 2 los siguientes LSAs:

- Router-LSA de r6 , describiendo sus interfaces r6-eth1, r6-eth2
- Router-LSA de r7 , describiendo todas sus interfaces.
- Router-LSA de r8 , describiendo todas sus interfaces.
- Router-LSA de r9 , describiendo todas sus interfaces.
- Network-LSA de la red 18.0.0.0/24 (creado por r6 , DR)
- Network-LSA de la red 19.0.0.0/24 (creado por r6 , DR)

- **Network-LSA** de la red 20.0.0.0/24 (creado por **r9** , DR)
- **Network-LSA** de la red 21.0.0.0/24 (creado por **r9** , DR)

En el área 2 se habrán recibido y propagado dentro del área los siguientes LSAs, estos LSAs se corresponden con información de subredes de otras áreas diferentes al área 2:

- **Summary-LSA** de la red 11.0.0.0/24 (creado por **r6**)
- **Summary-LSA** de la red 12.0.0.0/24 (creado por **r6**)
- **Summary-LSA** de la red 13.0.0.0/24 (creado por **r6**)
- **Summary-LSA** de la red 14.0.0.0/24 (creado por **r6**)
- **Summary-LSA** de la red 15.0.0.0/24 (creado por **r6**)
- **Summary-LSA** de la red 16.0.0.0/24 (creado por **r6**)
- **Summary-LSA** de la red 17.0.0.0/24 (creado por **r6**)
- **Summary-LSA** de la red 22.0.0.0/24 (creado por **r6**)

Resumen de mensajes OSPF

- **HELLO**: Descubrimiento de vecinos, elección DR/BDR.
- **DB DESCRIPTION**: Intercambio inicial de información relacionada con las *link-state databases*.
 - Cada *router* envía a un vecino (unicast) cuáles son los LSAs almacenados en sus DBs, especificando el tipo de LSA, el identificador del *router* que envió el LSA y el número de secuencia del LSA pero no el contenido del LSA.
- **LS REQUEST**: Petición a un vecino (unicast) de los LSAs que un *router* no tiene en sus DBs.
- **LS UPDATE**: mensaje que contiene uno o varios LSAs: **Router-LSA** , **Network-LSA** y **Summary-LSA** (hay otros).
- **LS ACK**: Los LSA contenidos en un LSU se asientan con un mensaje **LS ACK** .

Referencias

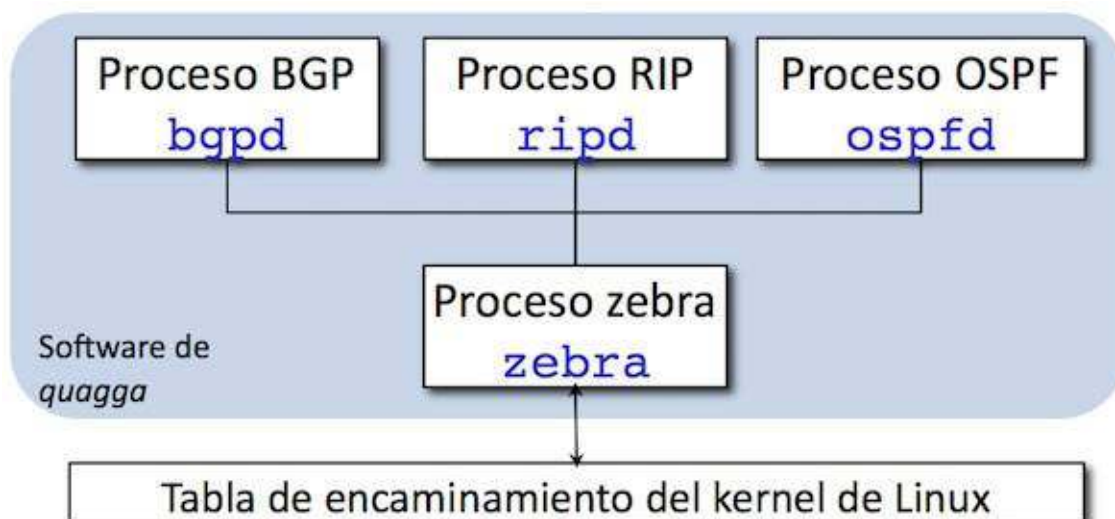
- Charles M. Kozierok, **TCP/IP GUIDE. A Comprehensive, Illustrated Internet Protocols Reference**, No Starch Press, 2005: capítulo 39
(http://www.tcpipguide.com/free/t_OpenShortestPathFirstOSPF.htm)
- John T. Moy, **OSPF: Anatomy of an Internet Routing Protocol**, Addison-Wesley (Safari Books Online), 1998: capítulo 4.
- RFC 2328, **OSPF version 2**: <http://www.faqs.org/rfcs/rfc2328.html>

OSPF en Quagga

- Introducción
- Configuración y monitorización de los procesos de Quagga
- Ficheros de configuración
- Iniciar Quagga
- Monitorización de la configuración
 - Tabla de encaminamiento OSPF
 - Información de los vecinos OSP
 - Router Link State DB
 - Network Link State DB
 - Summary Link State DB
 - Resumen de las DBs

Quagga

- Quagga (www.quagga.net) es un software que gestiona la tabla de encaminamiento de una máquina Linux según el funcionamiento de varios protocolos de encaminamiento de la arquitectura TCP/IP.
- La arquitectura de Quagga está formada por un conjunto de procesos:
 - Proceso `zebra` : actualiza la tabla de encaminamiento e intercambia rutas según diferentes protocolos de encaminamiento
 - Proceso de cada protocolo de encaminamiento: `ripd` , `ospfd` , `bgpd`
- Utilizaremos Quagga para probar los protocolos: OSPFv2 y BGP-4.



Configuración y monitorización de los procesos de Quagga

- Configuración a través de los ficheros:
 - `daemons`
 - `ospfd.conf`
- Monitorización a través de:
 - capturas de tráfico, utilizando `tcpdump` con la opción `-s 0` que permite capturar los paquetes completos.
 - Shell VTY (Virtual Terminal Interface) : `vttysh` (págs. [vttysh-begin]–[vttysh-end]) La Shell VTY se comunica con cada uno de los procesos quagga de la máquina y permite configurar los protocolos de encaminamiento y monitorizar su comportamiento.

Ficheros de configuración

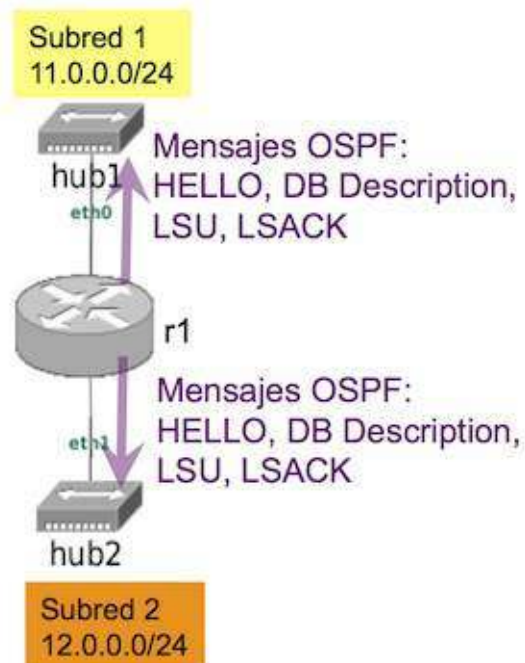
daemons

```
# Entries are in the format: <daemon>=(yes|no|priority)
# ...
# ...
# /usr/doc/quagga/README.Debian for details.
# Daemons are: bgpd quagga ospfd ospf6d ripd ripngd isisd
zebra=yes
bgpd=no
ospfd=yes
ospf6d=no
ripd=no
ripngd=no
isisd=no
```

Las líneas que comienzan por `#` son comentarios.

ospfd.conf

La configuración del fichero `ospfd.conf` en el router `r1` :



```
!
! OSPFd sample configuration file
!
hostname ospfd
password zebra

router ospf
router-id 12.0.0.1
network 11.0.0.0/24 area 0.0.0.0
network 12.0.0.0/24 area 0.0.0.0
```

Asignamos como ID del router la mayor de sus IPs por las que se activará OSPF:

```
router-id 12.0.0.1
```

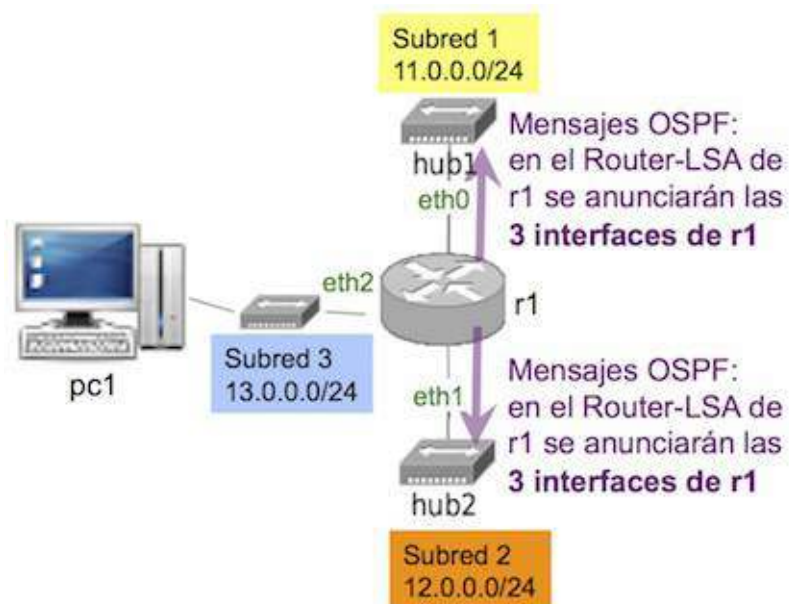
Se activa OSPF en las interfaces conectadas a estas redes, pertenecientes al área 0: a través de eth0 y eth1 se anunciarán las rutas utilizando OSPF. Hay que especificar a qué área pertenece cada interfaz del router por la que se activa OSPF.

```
network 11.0.0.0/24 area 0.0.0.0
network 12.0.0.0/24 area 0.0.0.0
```

Las líneas que comienzan por `!` son comentarios.

`ospfd.conf` : interfaces pasivas

La configuración del fichero `ospfd.conf` en el router `r1` con una interfaz pasiva:



```
! *- ospf *-
!
! OSPFd sample configuration file
!
hostname ospfd
password zebra

router ospf
router-id 13.0.0.1
passive-interface eth2
network 11.0.0.0/24 area 0.0.0.0
network 12.0.0.0/24 area 0.0.0.0
network 13.0.0.0/24 area 0.0.0.0
```

La interfaz `eth2` es pasiva, no se envían mensajes OSPF a través de ella.

Iniciar Quagga

- Al iniciar un *router* en NetGUI normalmente el software de quagga no estará arrancado. Para realizar una configuración:
 - Se editan los ficheros de configuración (págs [conf-begin]–[conf-end]).
 - Se arranca quagga: `/etc/init.d/quagga start`
 - Se realizar la monitorización.
 - Si es necesario modificar la configuración, se interrumpe la ejecución de quagga (`/etc/init.d/quagga stop`), se modifican los ficheros, y se vuelve a arrancar quagga (`/etc/init.d/quagga start`)

- En algunos escenarios puede que algunos *routers* estén preconfigurados para que arranquen con quagga ya lanzado.

Monitorización de la configuración: vtysh

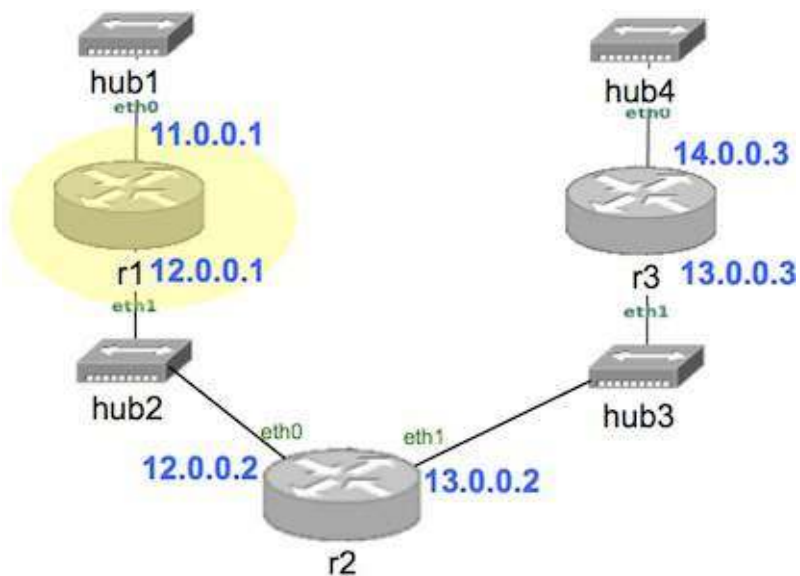
```
r1:~# vtysh
Copyright 1996-2005 Kunihiro Ishiguro, et al.

r1# ?
clear          Reset functions
configure     Configuration from vty interface
copy          Copy from one file to another
debug         Debugging functions (see also 'undebug')
disable       Turn off privileged mode command
end           End current mode and change to enable mode
exit          Exit current mde an down to previous mode
list          Print command list
no            Negate a command or set its defaults
ping          Send echo messages
quit          Exit current mode and down to previous mode
show          Show running system information
ssh           Open an ssh connection
start-shell   Start UNIX shell
telnet        Open a telnet connection
terminal      Set terminal line parameters
traceroute    Trace route to destination
undebug       Disable debugging functions (see also 'debug')
write         Write running configuration to memory, network, or terminal
r1#
```

Tabla de encaminamiento OSPF

Tabla de encaminamiento OSPF si sólo hay 1 área

- El comando `show ip ospf route` muestra la información sobre la tabla de encaminamiento OSPF del *router* (el ejemplo muestra la configuración del *router* `r1` de la figura):



```

r1# show ip ospf route
===== OSPF network routing table =====
N   11.0.0.0/24      [10] area: 0.0.0.0
      directly attached to eth0
N   12.0.0.0/24      [10] area: 0.0.0.0
      directly attached to eth1
N   13.0.0.0/24      [20] area: 0.0.0.0
      via 12.0.0.2, eth1
N   14.0.0.0/24      [30] area: 0.0.0.0
      via 12.0.0.2, eth1
===== OSPF router routing table =====
===== OSPF external routing table =====
    
```

Tabla de encaminamiento OSPF si hay varias áreas

- El comando `show ip ospf route` muestra información adicional:

```
r1# show ip ospf route
===== OSPF network routing table =====
N    15.0.0.0/24          [10] area: 0.0.0.1
                        directly attached to eth0
N    16.0.0.0/24          [10] area: 0.0.0.1
                        directly attached to eth2
N    17.0.0.0/24          [20] area: 0.0.0.1
                        via 16.0.0.5, eth2
N IA 18.0.0.0/24          [30] area: 0.0.0.1
                        via 16.0.0.5, eth2
N IA 19.0.0.0/24          [40] area: 0.0.0.1
                        via 16.0.0.5, eth2
N IA 20.0.0.0/24          [40] area: 0.0.0.1
                        via 16.0.0.5, eth2

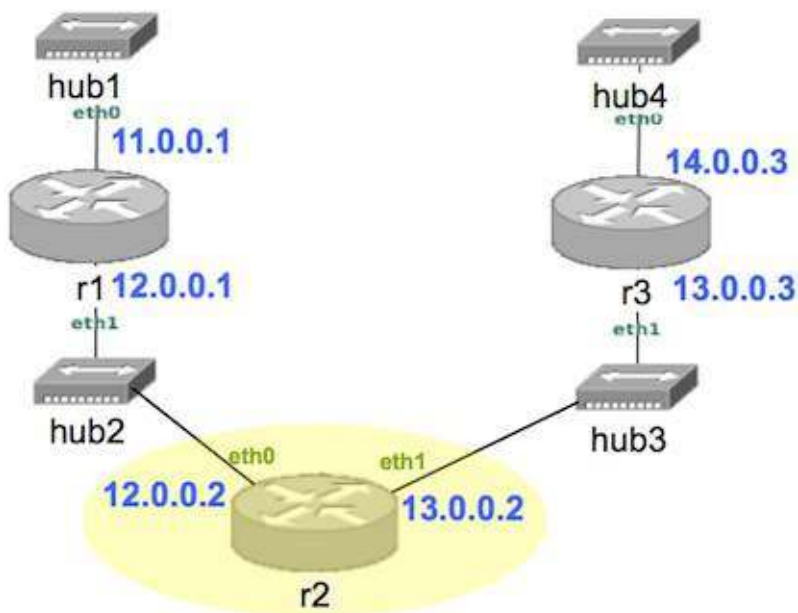
===== OSPF router routing table =====
R    18.0.0.4             [20] area: 0.0.0.1, ABR
                        via 16.0.0.5, eth2

===== OSPF external routing table =====
```

- ABR: router frontera de área
- IA: rutas inter-área (de otras áreas)
- El campo área indica a través de que área ha aprendido un router dicha ruta.
- Las rutas precedidas por N son rutas hacia una red.
- Las rutas precedidas por R son rutas hacia un *router*.

Información de los vecinos OSPF

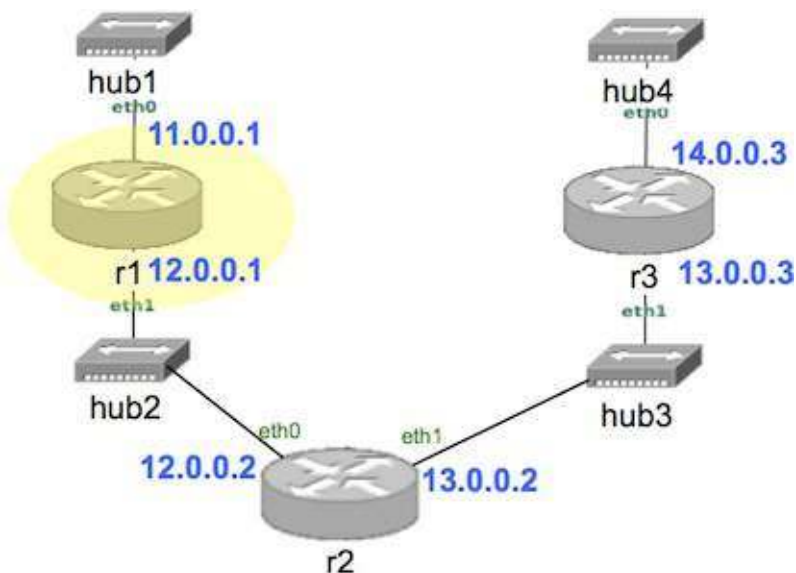
- El comando `show ip ospf neighbor` muestra la información sobre los vecinos que conoce el *router* (el ejemplo muestra el resultado del comando en el *router* `r2` de la figura):



```
r2# show ip ospf neighbor
Neighbor ID    Pri  State           Dead Time   Address      Interface
12.0.0.1      1    Full/Backup     00:00:30   12.0.0.1    eth0:12.0.0.2
14.0.0.3      1    Full/DR         00:00:40   13.0.0.3    eth1:13.0.0.2
```

Router Link State DB

- El comando `show ip ospf database router` muestra la información sobre la base de datos de *Router Link States* que conoce el *router* (el ejemplo muestra el resultado del comando en el *router* r1 de la figura):



```
r1# show ip ospf database router

OSPF Router with ID (12.0.0.1)

Router Link States (Area 0.0.0.0)
```



```
LS age: 1112
Options: 2
Flags: 0x0
LS Type: router-LSA
Link State ID: 12.0.0.1
Advertising Router: 12.0.0.1
LS Seq Number: 80000004
Checksum: 0x549d
Length: 48
  Number of Links: 2
```

```
  Link connected to: Stub Network
    (Link ID) Net: 11.0.0.0
    (Link Data) Network Mask: 255.255.255.0
      Number of TOS metrics: 0
        TOS 0 Metric: 10
```

```
  Link connected to: a Transit Network
    (Link ID) Designated Router address: 12.0.0.2
    (Link Data) Router Interface address: 12.0.0.1
      Number of TOS metrics: 0
        TOS 0 Metric: 10
```

```
LS age: 1107
Options: 2
Flags: 0x0
LS Type: router-LSA
Link State ID: 13.0.0.2
Advertising Router: 13.0.0.2
LS Seq Number: 80000004
Checksum: 0x2ab0
Length: 48
  Number of Links: 2
```

```
  Link connected to: a Transit Network
    (Link ID) Designated Router address: 12.0.0.2
    (Link Data) Router Interface address: 12.0.0.2
      Number of TOS metrics: 0
        TOS 0 Metric: 10
```

```
  Link connected to: a Transit Network
    (Link ID) Designated Router address: 13.0.0.3
    (Link Data) Router Interface address: 13.0.0.2
      Number of TOS metrics: 0
        TOS 0 Metric: 10
```

```
LS age: 1107
Options: 2
Flags: 0x0
LS Type: router-LSA
Link State ID: 14.0.0.3
Advertising Router: 14.0.0.3
```

```
LS Seq Number: 80000003
```

```
Checksum: 0xd210
```

```
Length: 48
```

```
Number of Links: 2
```

```
Link connected to: a Transit Network
```

```
(Link ID) Designated Router address: 13.0.0.3
```

```
(Link Data) Router Interface address: 13.0.0.3
```

```
Number of TOS metrics: 0
```

```
TOS 0 Metric: 10
```

```
Link connected to: Stub Network
```

```
(Link ID) Net: 14.0.0.0
```

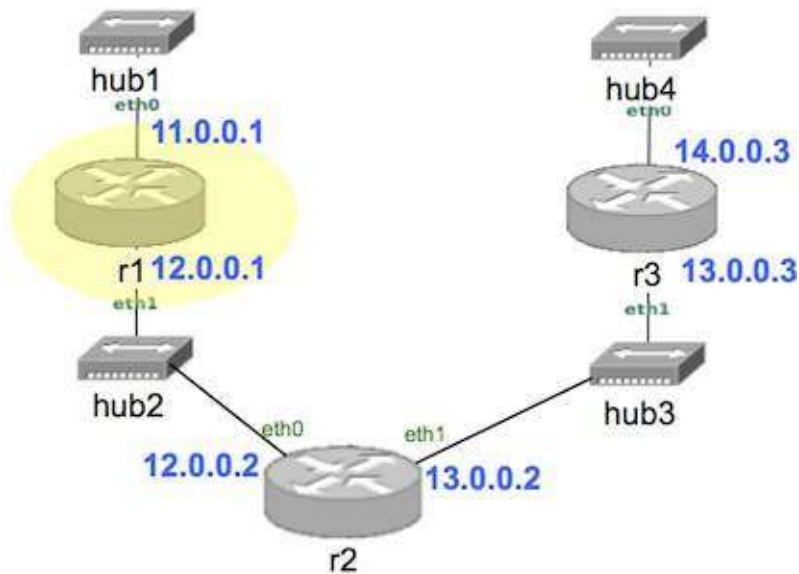
```
(Link Data) Network Mask: 255.255.255.0
```

```
Number of TOS metrics: 0
```

```
TOS 0 Metric: 10
```

Network Link State DB

- El comando `show ip ospf database network` muestra la información sobre la base de datos de *Network Link States* que conoce el *router* (el ejemplo muestra el resultado del comando en el *router* `r1` de la figura):



```
r1# show ip ospf database network

      OSPF Router with ID (12.0.0.1)

          Net Link States (Area 0.0.0.0)

LS age: 112
Options: 2
LS Type: network-LSA
Link State ID: 12.0.0.2 (address of Designated Router)
Advertising Router: 13.0.0.2
LS Seq Number: 80000002
Checksum: 0x5bc8
Length: 32
Network Mask: /24
    Attached Router: 12.0.0.1
    Attached Router: 13.0.0.2

LS age: 105
Options: 2
LS Type: network-LSA}
Link State ID: 13.0.0.3 (address of Designated Router)
Advertising Router: 14.0.0.3
LS Seq Number: 80000002
Checksum: 0x5fbc
Length: 32
Network Mask: /24
    Attached Router: 13.0.0.2
    Attached Router: 14.0.0.3
```

Summary Link State DB

- El comando `show ip ospf database summary` muestra la información sobre la base de datos de *Summary Link States* que conoce el *router*:

```
r1# show ip ospf database summary
      OSPF Router with ID (16.0.0.1)

          Summary Link States (Area 0.0.0.1)

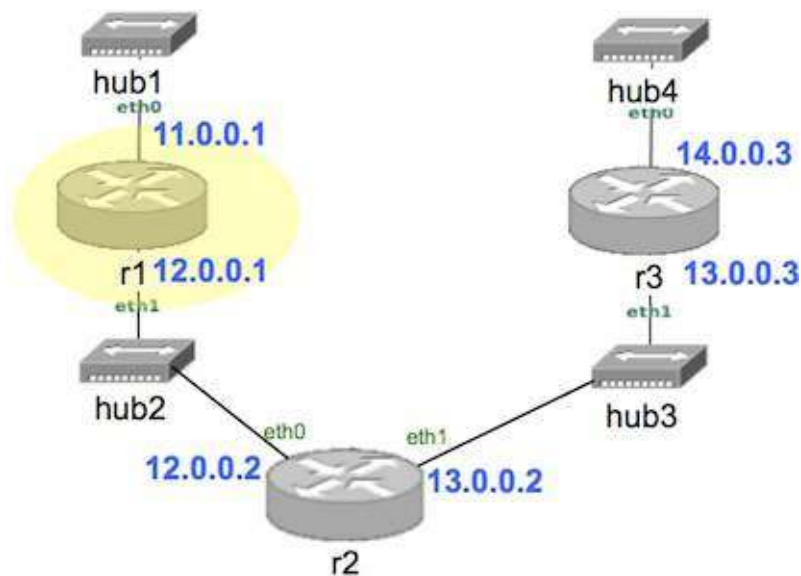
LS age: 592
Options: 2
LS Type: summary-LSA
Link State ID: 18.0.0.0 (summary Network Number)
Advertising Router: 18.0.0.4
LS Seq Number: 80000006
Checksum: 0x0c07
Length: 28
Network Mask: /24
      TOS: 0  Metric: 10

LS age: 580
Options: 2
LS Type: summary-LSA
Link State ID: 19.0.0.0 (summary Network Number)
Advertising Router: 18.0.0.4
LS Seq Number: 80000005
Checksum: 0x63a4
Length: 28
Network Mask: /24
      TOS: 0  Metric: 20

LS age: 588
Options: 2
LS Type: summary-LSA
Link State ID: 20.0.0.0 (summary Network Number)
Advertising Router: 18.0.0.4
LS Seq Number: 80000006
Checksum: 0x56b0
Length: 28
Network Mask: /24
      TOS: 0  Metric: 20
```

Resumen de las DBs si sólo hay 1 área

- El comando `show ip ospf database` muestra un resumen de la información sobre las bases de datos del *router* (el ejemplo muestra el resultado del comando en el *router* `r1` de la figura):



```
r1# show ip ospf database

      OSPF Router with ID (12.0.0.1)

      Router Link States (Area 0.0.0.0)

Link ID        ADV Router    Age  Seq#       CkSum  Link count
-----
12.0.0.1      12.0.0.1     579  0x80000005 0x529e    2
13.0.0.2      13.0.0.2     574  0x80000005 0x28b1    2
14.0.0.3      14.0.0.3     574  0x80000004 0xd011    2

      Net Link States (Area 0.0.0.0)

Link ID        ADV Router    Age  Seq#       CkSum
-----
12.0.0.2      13.0.0.2     586  0x80000002 0x5bc8
13.0.0.3      14.0.0.3     579  0x80000002 0x5fbc
```

Resumen de las DBs si hay varias áreas

- El comando `show ip ospf database` también muestra la información de los *Summary Link States*:

```
r1# show ip ospf database
```

```
OSPF Router with ID (12.0.0.1)
```

```
Router Link States (Area 0.0.0.1)
```

Link ID	ADV Router	Age	Seq#	CkSum	Link count
12.0.0.1	12.0.0.1	579	0x80000005	0x529e	2
13.0.0.2	13.0.0.2	574	0x80000005	0x28b1	2
14.0.0.3	14.0.0.3	574	0x80000004	0xd011	2

```
Net Link States (Area 0.0.0.1)
```

Link ID	ADV Router	Age	Seq#	CkSum
12.0.0.2	13.0.0.2	586	0x80000002	0x5bc8
13.0.0.3	14.0.0.3	579	0x80000002	0x5fbc

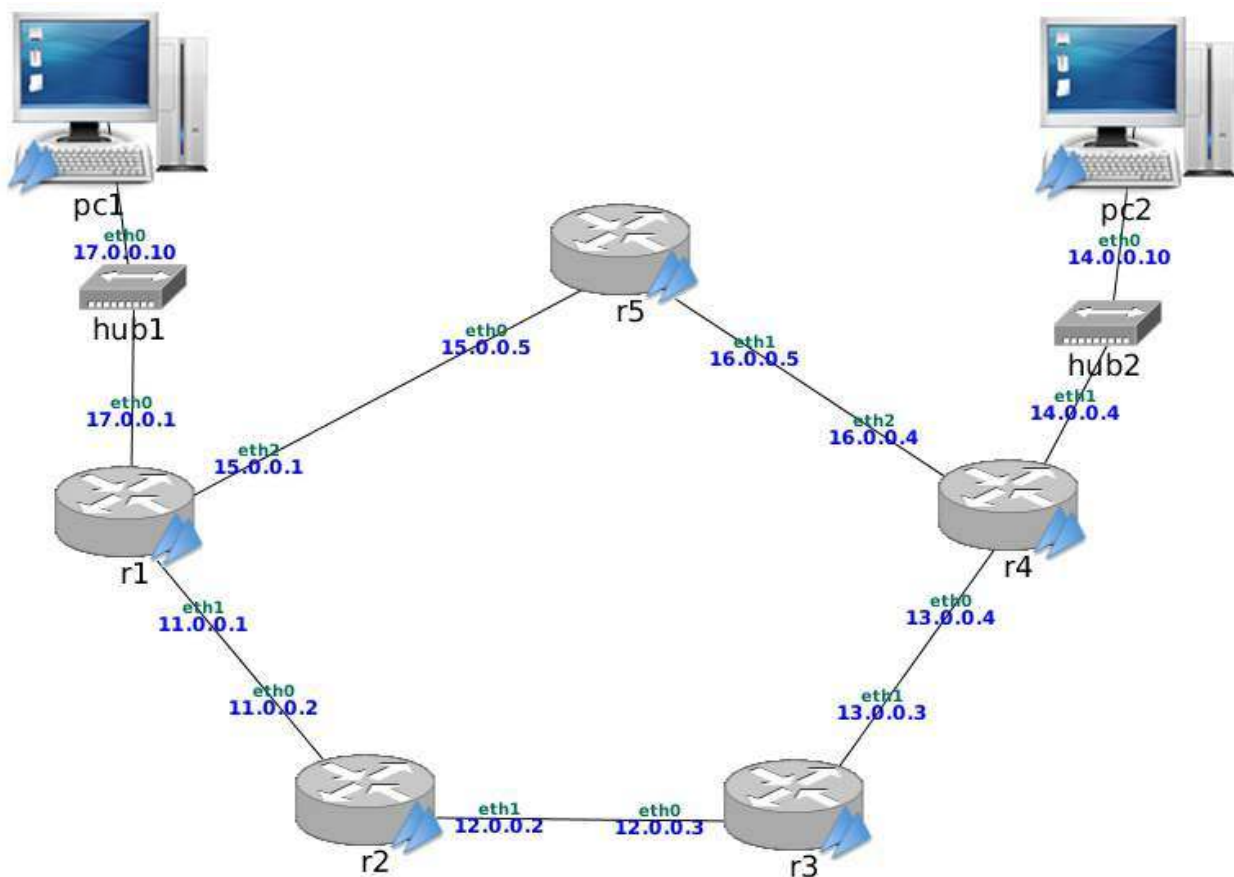
```
Summary Link States (Area 0.0.0.1)
```

Link ID	ADV Router	Age	Seq#	CkSum	Route
18.0.0.0	18.0.0.4	592	0x80000006	0x0c07	18.0.0.0/24
19.0.0.0	18.0.0.4	580	0x80000005	0x61a5	19.0.0.0/24
20.0.0.0	18.0.0.4	588	0x80000006	0x56b0	20.0.0.0/24

Laboratorio de OSPF

- Red con una única área
 - Introducción
 - Activación de r1
 - Activación de r2
 - Activación de r3 y r4
 - Activación y desactivación de r5
 - Borrado de rutas
- Red con varias áreas

Red con una única área



Introducción

1. En el fichero `lab-OSPF.tgz` está definida una red como la que se muestra en la figura [fig:ospf1]. Descomprime el fichero de configuración del escenario `lab-OSPF.tgz`. Al arrancar NetGUI debes abrir el escenario definido en el directorio `lab-OSPF`.
2. Arranca todas las máquinas de una en una. Las máquinas `pc1` y `pc2` tienen rutas por defecto a `r1` y `r4` respectivamente. Los *routers* no tienen configurada ninguna ruta, salvo la de las subredes a las que están directamente conectados. Compruébalo con la orden `route`. En los siguientes apartados se configurará OSPF en cada *router* para que las tablas de encaminamiento permitan alcanzar cualquier punto de la red

Activación de `r1`

Para observar los mensajes que envíe `r1` cuando se active OSPF, arranca `tcpdump` en `pc1`, en `r2(eth0)` y en `r5(eth0)` utilizando la opción `-s 0` para que capture los paquetes completos y guardando la captura en un fichero con la opción `-w`.

A continuación configura OSPF en el encaminador `r1` para que su identificador de *router* sea la mayor de sus direcciones IP y para que exporte las rutas hacia las tres redes a las que está conectado. Para ello edita los ficheros `daemons` y `ospfd.conf`, y después arranca *quagga*. Debes configurar **todos** los *routers* de la figura dentro del mismo área, con identificador de área igual a 0.

Espera un minuto aproximadamente e interrumpe las capturas.

Analiza el comportamiento de `r1` estudiando las capturas con *wireshark* y consultando el estado de OSPF a través de su interfaz VTY y de la orden `route`:

1. Observa los mensajes `HELLO` que se envían al arrancar *quagga* en `r1` y analízalos utilizando Wireshark.
 - i. ¿Cada cuanto tiempo se envían dichos mensajes? Observa si coincide con el valor del campo `Hello Interval` de los mensajes.
 - ii. Comprueba que el campo `Area ID` se corresponde con el identificador de área que has configurado en el fichero `ospfd.conf`.
 - iii. Comprueba que el identificador del *router* se corresponde con el que has configurado en el fichero mirando el campo `Source OSPF Router` de la cabecera obligatoria de OSPF en los mensajes `HELLO`.

Comprueba que este identificador es el mismo para los mensajes enviados por cualquiera de las interfaces de `r1`, aunque los mensajes se envíen con dirección IP origen diferente (cada mensaje llevará como dirección IP origen la de la interfaz de red de `r1` por la que se envíe).

- iv. Observa el valor de los campos `DR` y `BDR` en los primeros mensajes `HELLO`.
¿Qué ocurre con dichos campos transcurridos 40 segundos después del primer mensaje `HELLO`? ¿Por qué?
2. ¿Se observan en las capturas mensajes `DB Description` o `LS Update`? ¿Por qué?
3. ¿Debería haber aprendido alguna ruta `r1`? Compruébalo consultando la tabla de encaminamiento mediante la orden `route`.
4. Consulta la información de OSPF relativa a la tabla de encaminamiento utilizando la interfaz VTY en `r1` con `show ip ospf route`.
5. Consulta la información de los vecinos que ha conocido `r1` a través de los mensajes `HELLO` recibidos mediante `show ip ospf neighbor`.
6. Consulta la información de la base de datos de *Router Link States* de `r1` con `show ip ospf database router`.
7. Consulta la información de la base de datos de *Network Link States* de `r1` con `show ip ospf database network`

Activación de `r2`

Para observar los mensajes que envíe `r2` cuando se active OSPF, y los que envíe `r1` a consecuencia de la activación de `r2`, arranca `tcpdump` en `r1(eth1)`, en `r3(eth0)` y en `r5(eth0)` utilizando la opción `-s 0` para que capture los paquetes completos y guardando la captura en un fichero con la opción `-w`.

A continuación configura OSPF en el encaminador `r2` para que su identificador de *router* sea la mayor de sus direcciones IP y para que exporte las rutas hacia las dos redes a las que está conectado. Para ello edita los ficheros `daemons` y `ospfd.conf`, y después arranca *quagga*.

Espera dos minutos aproximadamente e interrumpe las capturas.

Analiza el comportamiento de `r2` y `r1` estudiando las capturas con *wireshark* y consultando el estado de OSPF a través de las interfaces VTY y de la orden `route` en cada encaminador:

1. Observa la captura realizada en `r1` y responde a las siguientes cuestiones:
 - i. ¿Qué tipo de mensajes aparecen cuando `r1` detecta la presencia de `r2` y viceversa? ¿Cuál es su propósito? ¿Qué IP de destino llevan esos mensajes?

ii. Observa los mensajes `LS Request` que envían `r1` y `r2`. ¿Qué `LSA` pide cada uno al otro? ¿Qué IP de destino llevan estos mensajes?

iii. Observa el primer mensaje `LS Update` que envía `r1`. Comprueba que se corresponde con el `LS Request` enviado por `r2`. Comprueba cómo se corresponde su contenido con lo almacenado en la base de datos de `r1` analizada en el apartado anterior. Observa sus campos para ver si este mensaje incluye la información de que `r1` ha descubierto a `r2` como vecino. ¿Crees que la información contenida en este mensaje deberá cambiar próximamente? ¿Por qué?

Observa el campo `LS Age` del anuncio que viaja en el mensaje, y explica su valor.

iv. Observa el primer mensaje `LS Update` que envía `r2`. Comprueba que se corresponde con el `LS Request` enviado por `r1`. Observa sus campos para ver si este mensaje incluye la información de que `r2` ha descubierto a `r1` como vecino. ¿Crees que la información contenida en este mensaje deberá cambiar próximamente? ¿Por qué?

Observa el campo `LS Age` del anuncio que viaja en el mensaje, y explica su valor.

v. Observa el segundo y tercer mensajes `LS Update` que envía `r1`. ¿Responden a algún `LS Request` previo? ¿Por qué se envían? ¿Qué información contienen?

Observa el campo `LS Age` de los anuncios que viajan en los mensajes, y explica su valor.

vi. Observa el segundo mensaje `LS Update` que envía `r2`. ¿Responde a algún `LS Request` previo? ¿Por qué se envía? ¿Qué información contiene?

Observa el campo `LS Age` del anuncio que viaja en el mensaje, y explica su valor.

vii. ¿Por qué razón `r2` no envía ningún mensaje `Network-LSA` ?

viii. Observa los mensajes `LS Acknowledge`. Mira su contenido para comprobar a qué `LSA` s asienten.

ix. Pasados 40 segundos del arranque de `r2`, ¿qué ocurre con los campos `DR` y `BDR` de los mensajes `HELLO` que intercambian?

2. Observa la captura realizada en `r5` :

i. Explica por qué no aparecen los mensajes `LS Update` que crea `r1` y envía a `r2`.

ii. Explica por qué no aparecen los mensajes `LS Update` que crea `r2` y envía a `r1`, y `r1` debería propagar por inundación.

3. Observa la captura realizada en `r3` :
 - i. Explica por qué no aparecen los mensajes `LS Update` que crea `r2` y envía a `r1` .
 - ii. Explica por qué no aparecen los mensajes `LS Update` que crea `r1` y envía a `r2` , y `r2` debería propagar por inundación.
4. ¿Deberían haber aprendido alguna ruta `r2` y `r1` ? Compruébalo consultando la tabla de encaminamiento en ambos encaminadores mediante la orden `route` .
5. Consulta la información de OSPF relativa a la tabla de encaminamiento utilizando la interfaz VTY en cada encaminador con `show ip ospf route` . Comprueba la métrica de cada ruta y a través de qué *router* se alcanza.
6. Consulta la información de los vecinos que ha conocido cada encaminador a través de los mensajes `HELLO` mediante `show ip ospf neighbor` .
7. Consulta en cada encaminador la información de las bases de datos de *Router Link States* y de *Network Link States* mediante `show ip ospf database router` y `show ip ospf database network` respectivamente.

Comprueba que la información mostrada coincide con el contenido de los últimos `LS update` enviados por los encaminadores.
8. Consulta un resumen de las bases de datos en cada encaminador con `show ip ospf database` .

Activación de `r3` y `r4`

Para observar los mensajes que envíen `r3` y `r4` cuando activen OSPF, y los que envíe `r2` a consecuencia de la activación de `r3` y `r4` , arranca `tcpdump` en `r1(eth1)` , en `r2(eth1)` y en `r3(eth1)` utilizando la opción `-s 0` para que capture los paquetes completos y guardando la captura en un fichero con la opción `-w` .

Configura OSPF en `r3` y en `r4` , y **arráncalos a la vez**. Analiza el comportamiento de los encaminadores estudiando las capturas con *wireshark* y consultando el estado de OSPF a través de las interfaces VTY y de la orden `route` en cada encaminador:

1. Trata de suponer los valores de `DR` y `BDR` en las subredes `12.0.0.0/24` y `13.0.0.0/24` . Comprueba si tus suposiciones son ciertas. Comprueba en los mensajes `HELLO` de la captura en `r3` cómo se ha producido la elección de `DR` y `BDR` al arrancar `r3` y `r4` a la vez.

2. En la captura en `r3` observa el intercambio de mensajes `LS Update` que se produce mientras arrancan `r3` y `r4`.
3. En la captura en `r2` observa el intercambio de mensajes `LS Update` que se produce mientras arrancan `r3` y `r4`.

Observa también en dicha captura los mensajes `LS Update` que `r3` envía por inundación de los recibidos por él de `r4`. Indica cómo puedes saber si un `LS Update` lo ha originado el encaminador que lo envía o está siendo propagado por inundación.
4. Antes de examinar la captura en `r1` trata de suponer qué tipos de mensaje aparecerán en ella. Comprueba tus suposiciones.
5. Trata de suponer qué modificaciones se habrán realizado en las tablas de encaminamiento de cada *router*. Observa las tablas de encaminamiento utilizando la interfaz VTY con el proceso `ospfd` para verificar tus suposiciones.
6. Consulta la información de los vecinos que ha conocido cada encaminador a través de los mensajes `HELLO` mediante la interfaz VTY.
7. Consulta en cada encaminador la información de las bases de datos de *Router Link States* y de *Network Link States*.

Comprueba que la información mostrada coincide con el contenido de los últimos `LS update` enviados por los encaminadores.
8. Consulta el resumen de las bases de datos en cada encaminador.

Reconfiguración de rutas:\

Activación y desactivación de `r5`

Después de realizar los apartados anteriores, OSPF está funcionando en los encaminadores `r1`, `r2`, `r3` y `r4`. Por tanto, `pc1` y `pc2` deberían tener conectividad IP. Compruébalo con las órdenes `ping` y `traceroute`.

1. Interrumpe *quagga* en los encaminadores `r1`, `r2`, `r3` y `r4`. Comprueba que ya no funciona un `ping` de `pc1` a `pc2`. Deja lanzado el `ping` de `pc1` a `pc2`, y reanuncia *quagga* en `r1`, `r2`, `r3`, `r4`, fijándote en los segundos (aproximadamente) que pasan desde que está arrancado *quagga* en todos los encaminadores hasta que el `ping` empieza a funcionar. Apunta este valor de tiempo.
2. Comprueba cómo ahora los valores de DR y BDR en las subredes de la figura pueden haber cambiado con respecto a lo que observaste en los apartados anteriores, ya que ahora *quagga* estará arrancando más o menos simultáneamente en todos. Indica

cuáles son los valores actuales de DR y BDR en cada subred.

3. Fíjate en la tabla de encaminamiento de `r1` que se muestra con la orden `route`. Fíjate en la métrica para la red `14.0.0.0/24`.
4. Realiza los cambios necesarios para que la ruta seguida por los datagramas IP que envía `pc1` a `pc2` vayan por la ruta `pc1 => r1 => r5 => r4 => pc2`, y para que los que envía `pc2` a `pc1` vayan por la ruta `pc2 => r4 => r5 => r1 => pc1`. Para realizar este apartado no podrás añadir o eliminar manualmente rutas en las tablas de encaminamiento. Mirando la tabla de encaminamiento de `r1`, observa y apunta el número de segundos que aproximadamente tarda en aprender `r1` la nueva ruta.

Comprueba que se está utilizando dicha ruta a través de la orden `traceroute`. Comprueba las rutas y sus métricas en las tablas de encaminamiento de cada encaminador.

Comprueba cómo ha mejorado la métrica para la red `14.0.0.0/24` desde el *router* `r1`.

5. Comprueba la ruta que están siguiendo los mensajes intercambiados entre `pc1` y `pc2` con `traceroute`.

Deja corriendo en `pc1` un `ping` hacia `pc2`.

6. A continuación interrumpe la ejecución de *quagga* en el encaminador `r5` utilizando la orden `/etc/init.d/quagga stop`. Podrás observar con la orden `route` que ahora `r5` no conoce rutas aprendidas por OSPF. Tampoco exporta información de vecinos hacia otros encaminadores.
7. Observarás que el `ping` de `pc1` a `pc2` deja de funcionar durante un buen rato (fíjate en el número de secuencia `icmp_seq`, éste aumenta con cada paquete enviado cada segundo).

Observa durante este periodo, en el que no está funcionando `r5`, la tabla de encaminamiento de `r1` y `r4`.

Observa también durante este periodo la lista de vecinos conocidos por `r1` y por `r4` (utilizando la interfaz VTY con el proceso `ospfd`). Observa la evolución de la columna `Dead Time` de las distintas entradas. ¿Qué entradas no reinician la cuenta desde los 40 segundos? ¿Por qué?

8. Espera hasta que vuelva a funcionar el `ping` (fíjate en el número `icmp_seq`). Observa y apunta el número de segundos que aproximadamente está sin funcionar el `ping` debido a que aún no se ha olvidado la ruta a través de `r5`. Comprueba que finalmente `r5` ha desaparecido de entre los vecinos conocidos por `r1` y `r4`.
9. Comprueba ahora las entradas de las tablas de encaminamiento de `r1` y de `r4`.

Interrumpe el `ping` y comprueba la ruta que están siguiendo los mensajes intercambiados entre `pc1` y `pc2` con `traceroute`.

10. Por último, vuelve a arrancar de nuevo `quagga` en `r5`. Observa cómo cambian las tablas de encaminamiento en `r1` y `r4` y apenas se interrumpe el `ping`.

Comprueba de nuevo cuál es ahora la ruta que están siguiendo los mensajes intercambiados entre `pc1` y `pc2` con `traceroute`. Observa y apunta el número de segundos que aproximadamente tarda en aprenderse de nuevo la ruta a través de `r5`, mirando continuamente la tabla de encaminamiento de `r1`. Mira también los números de secuencia de los `icmps` del `ping`, y fíjate si alguno se pierde mientras se cambia de la ruta antigua a la ruta nueva.

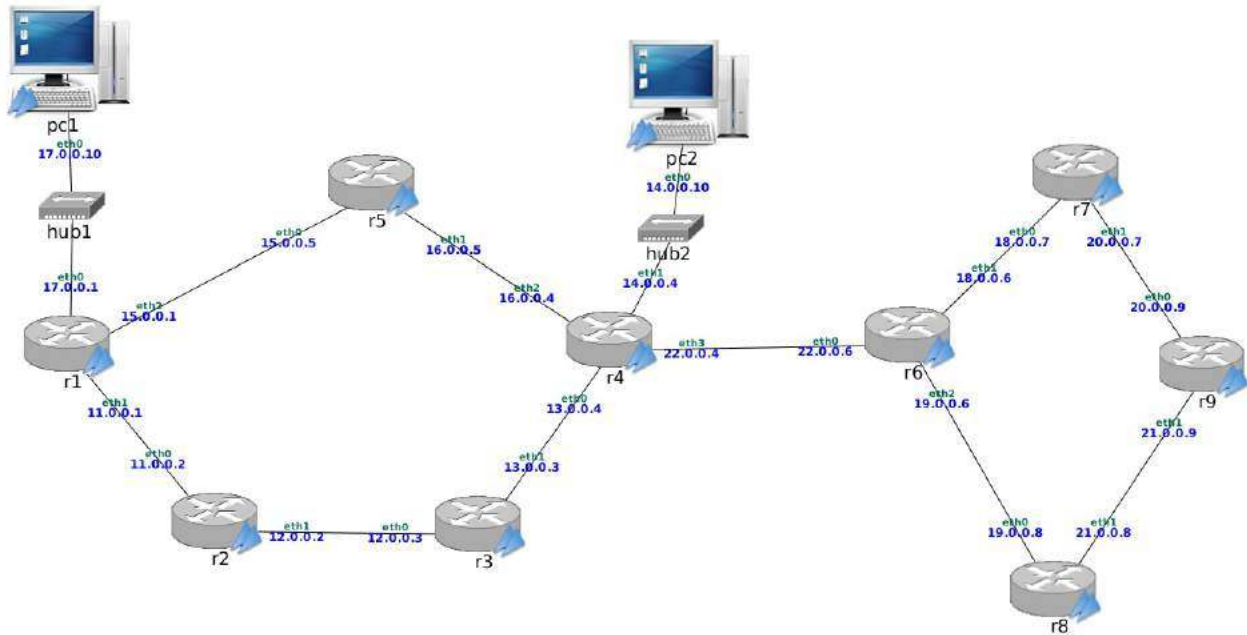
Borrado de rutas

Comprueba ahora qué ocurre cuando se interrumpe `quagga` en uno de los routers de la figura:

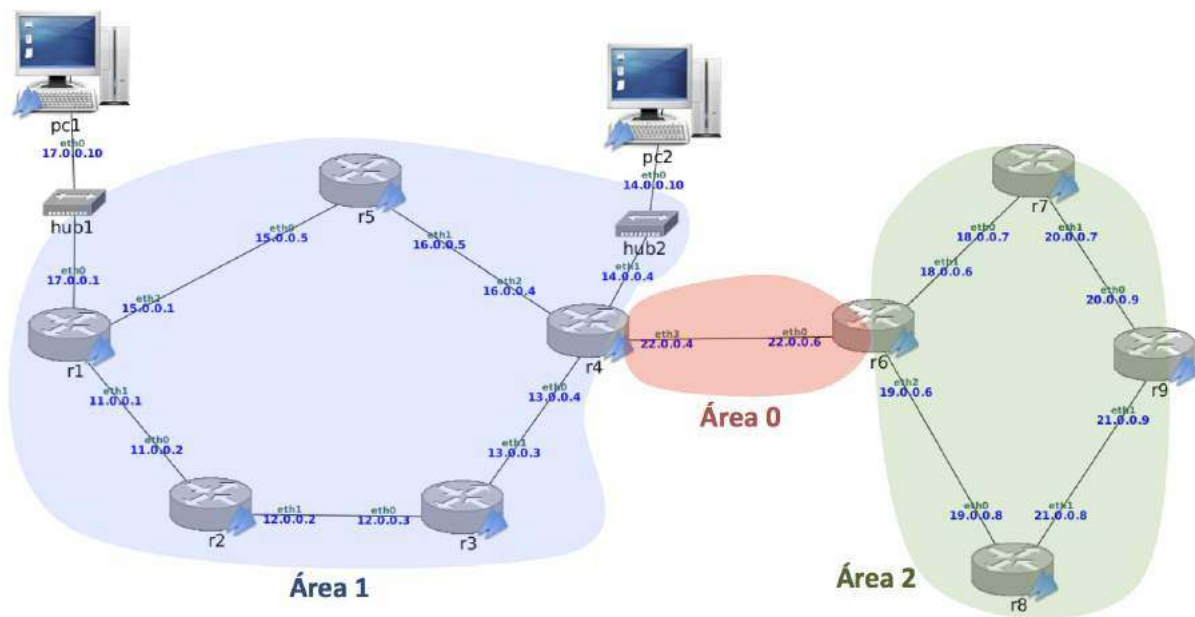
- Interrumpe `quagga` en los encaminadores `r1`, `r2`, `r3`, `r4`, y `r5`. Inicia `quagga` en los mismos de tal forma que el arranque sea aproximadamente simultáneo y observa los DR y BDR de cada subred.
- Arranca dos capturas de tráfico para que se capturen todos los paquetes de la subred 15.0.0.0/24 y de la subred 11.0.0.0/24. Interrumpe `quagga` en `r4`. Espera unos segundos hasta que en `r1` haya desaparecido la subred 14.0.0.0/24 e interrumpe las capturas de tráfico. Comprueba que no funciona un `ping` entre `pc1` y `pc2`. Observa cómo han cambiado las tablas de encaminamiento de los routers.
- Analiza las capturas de tráfico e indica que anuncios LSA ha recibido `r1` como resultado de haber apagado `r4`. Explica por qué crees que no hay un mensaje Network-LSA de la subred 13.0.0.0/24 y sí hay un mensaje Network-LSA de la subred 16.0.0.0/24. Observa con atención el campo LS-Age de este mensaje.
- Observa en las bases de datos de Router-LSA como el mensaje Router-LSA de `r4` sigue permaneciendo. ¿Cuándo crees que se borrará? ¿Cómo saben los routers de la figura que ha desaparecido `r4`?

Red con varias áreas

En el fichero `lab-OSPF-Areas.tgz` está definida una red como la que se muestra en la figura [fig:ospf2]. Descomprime el fichero de configuración del escenario `lab-OSPF-Areas.tgz`. Al arrancar NetGUI debes abrir el escenario definido en el directorio `lab-OSPF-Areas`.



- Arranca todas las máquinas de una en una. Las máquinas `pc1` y `pc2` tienen rutas por defecto a `r1` y `r4` respectivamente. Los *routers* tienen configurado OSPF, **estando todos ellos en el área 0**.
- Arranca *quagga* en todos los *routers*, y espera aproximadamente un minuto.
- Con la orden `route` comprueba las tablas de encamamiento en `r1`, `r4`, `r6` y `r9`. Deberían tener ruta a todas las redes de la figura. Comprueba el coste de cada ruta. [rutas]
- Comprueba en esos mismos *routers*, a través de su interfaz VTY, los mensajes LSU *Router-LSA* y *Network-LSA* presentes en sus bases de datos. Toma nota de qué mensajes hay exactamente. [databases]
- Apaga *quagga* en todos los *routers*. Configura ahora todos ellos de forma que se establezcan las áreas que se muestran en la figura [fig:ospf3]. Para ello, edita sus ficheros `/etc/quagga/ospfd.conf` y cambia el área al que pertenece cada interfaz de cada router en las líneas `network`.



- Reinicia *quagga* en todos los *routers* **excepto** r4 y r6 , y espera aproximadamente un minuto.
- Mira las bases de datos de r1 y r5 . ¿Hay algún mensaje LSU *Summary-LSA* en ellas? ¿Por qué?
- Para observar los mensajes que envíen r4 y r6 cuando activen OSPF, arranca *tcpdump* en r3(eth1) , r4(eth3) y r7(eth0) .
- Arranca ahora *quagga* en r4 y r6 , y espera aproximadamente un minuto. Interrumpe las capturas.
- Localiza en la captura los mensajes *LS Update* que envía r4 a r3 que permiten a r3 añadir una ruta para cada una de las siguientes redes:
 - 18.0.0.0/24
 - 19.0.0.0/24
 - 20.0.0.0/24
 - 21.0.0.0/24

¿De qué tipo de LSAs se trata?

- Localiza en la captura los mensajes *LS Update* que envía r6 a r7 que permiten a r7 añadir una ruta para cada una de las siguientes redes:
 - 11.0.0.0/24
 - 12.0.0.0/24
 - 13.0.0.0/24

- 14.0.0.0/24
- 15.0.0.0/24
- 16.0.0.0/24
- 17.0.0.0/24

¿De qué tipo de LSAs se trata?

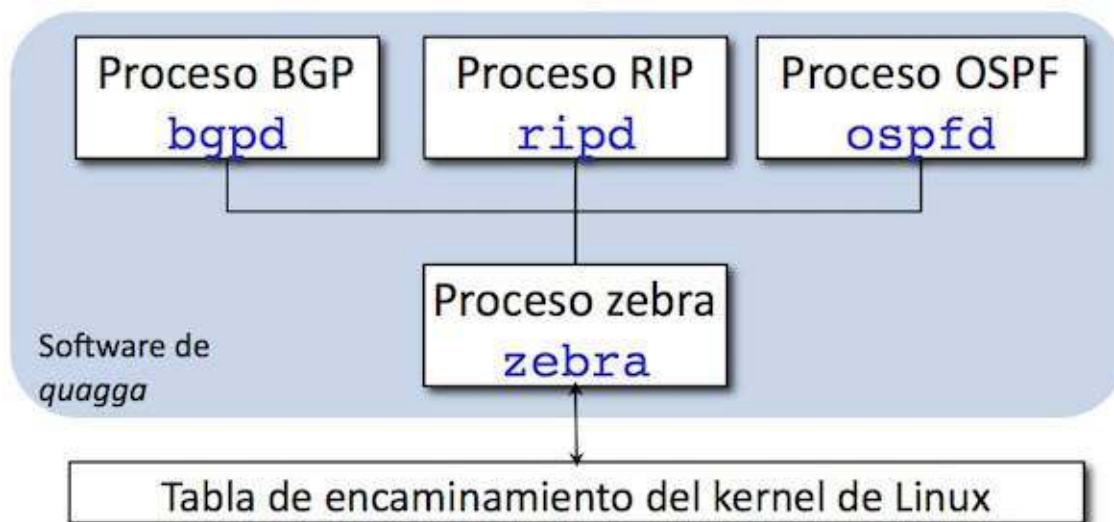
- Localiza en las tres capturas qué tipo de LSA contiene el anuncio de la existencia de la red 22.0.0.0/24 :
 - cuando r3 la aprende de r4
 - cuando r6 la aprende de r4
 - cuando r7 la aprende de r6
- Localiza en las tres capturas qué tipo de LSA contiene el anuncio de la existencia de la red 14.0.0.0/24 :
 - cuando r3 la aprende de r4
 - cuando r6 la aprende de r4
 - cuando r7 la aprende de r6
- Comprueba las tablas de encaminamiento en r1 , r4 , r6 y r9 . Comprueba el coste de cada ruta. Compara los resultados con los obtenidos en la pregunta [rutas].
- Comprueba en esos mismos *routers*, a través de su interfaz VTY, los mensajes LSU *Router-LSA*, los *Network-LSA* y los *Summary-LSA* presentes en sus bases de datos. Compara con los resultados obtenidos en la pregunta [databases].

BGP en Quagga

- [Introducción](#)
- [Configuración y monitorización de los procesos de Quagga](#)
- [Ficheros de configuración](#)
- [Iniciar Quagga](#)
- [Monitorización de la configuración](#)
 - [Tabla de encaminamiento BGP](#)
- [Redistribución de rutas entre BGP y OSPF](#)
- [Políticas de exportación de rutas](#)
- [Selección de la mejor ruta](#)
- [Configuración del atributo LOCAL PREF](#)
- [Configuración de una ruta por defecto](#)

Quagga

- Quagga (www.quagga.net) es un software que gestiona la tabla de encaminamiento de una máquina Linux según el funcionamiento de varios protocolos de encaminamiento de la arquitectura TCP/IP.
- La arquitectura de Quagga está formada por un conjunto de procesos:
 - Proceso `zebra` : actualiza la tabla de encaminamiento e intercambia rutas según diferentes protocolos de encaminamiento
 - Proceso de cada protocolo de encaminamiento: `ripd` , `ospfd` , `bgpd`
- Utilizaremos Quagga para probar los siguientes protocolos: RIPv2, OSPFv2 y BGP-4.



Configuración y monitorización de los procesos de Quagga

- Configuración a través de los ficheros:
 - `daemons`
 - `bgpd.conf`
- Monitorización a través de:
 - capturas de tráfico, utilizando `tcpdump` con la opción `-s 0` que permite capturar los paquetes completos.
 - Shell VTY (Virtual Terminal Interface) : `vtysh` (págs. [vtysh-begin]–[vtysh-end]) La Shell VTY se comunica con cada uno de los procesos quagga de la máquina y permite configurar los protocolos de enrutamiento y monitorizar su comportamiento.

Ficheros de configuración

`daemons`

```

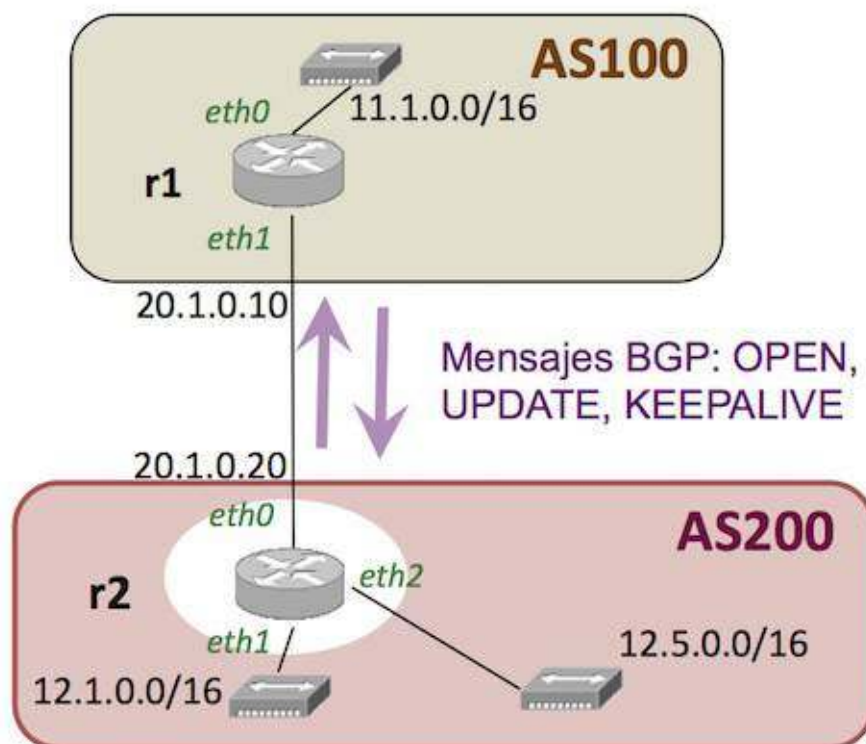
# Entries are in the format:  <daemon>=(yes|no|priority)
# ...
# ...
# /usr/doc/quagga/README.Debian for details.
# Daemons are: bgpd quagga ospfd ospf6d ripd ripngd isisd
zebra=yes
bgpd=yes
ospfd=no
ospf6d=no
ripd=no
ripngd=no
isisd=no

```

Las líneas que comienzan por `#` son comentarios.

bgpd.conf

Para que un router pueda establecer una sesión BGP deberá configurar su fichero `bgpd.conf`. A continuación se muestra la configuración de `r2`:



```
! *- bgpf *-  
!  
! BGPd sample configuration file  
!  
hostname bgpd  
password zebra  
  
router bgp 200  
bgp router-id 20.1.0.20  
  
neighbor 20.1.0.10 remote-as 100  
  
redistribute connected
```

Las líneas que comienzan por `!` son comentarios.

Es necesario indicar el número de sistema autónomo al que pertenece el router, en el ejemplo ASN=200 y su identificador (utilizaremos como convenido la dirección IP más alta del router):

```
router bgp 200  
bgp router-id 20.1.0.20
```

El router tendrá que especificar quién son los routers BGP vecinos con los que establecerá la sesión BGP, en este caso sólo hay un vecino 20.1.0.10 que pertenece al sistema autónomo número 100:

```
neighbor 20.1.0.10 remote-as 100
```

Se anunciarán por BGP las redes a las que el router está directamente conectado (en este caso: 20.1.0.0/16, 12.1.0.0/16, 12.5.0.0/16). Es equivalente a haber escrito:

```
network 20.1.0.0/16  
network 12.1.0.0/16  
network 12.5.0.0/16
```

bgpd.conf : Agregación de Rutas

Utilizando CIDR pueden agruparse varias redes bajo un solo prefijo para optimizar el número de entradas en las tablas de los *routers*.

En el fichero `bgpd.conf` se incluirá el comando:

```
aggregate-address a.b.c.d/prefix summary-only
```

si una red a anunciar se encuentra incluida en `a.b.c.d/prefix` , se anunciará `a.b.c.d/prefix` en vez de dicha red.

Ejemplo:

```
...
router bgp 200
...
    aggregate-address 12.0.0.0/14 summary-only
...
```

Agrega las redes 12.0.0.0/16, 12.1.0.0/16, 12.2.0.0/16, 12.3.0.0/16 bajo el prefijo 12.0.0.0/14.

Iniciar Quagga

- Al iniciar un *router* en NetGUI normalmente el software de quagga no estará arrancado. Para realizar una configuración:
 1. Se editan los ficheros de configuración (págs [conf-begin]–[conf-end]).
 2. Se arranca quagga: `/etc/init.d/quagga start`
 3. Se realizar la monitorización.
 4. Si es necesario modificar la configuración, se interrumpe la ejecución de quagga (`/etc/init.d/quagga stop`), se modifican los ficheros, y se vuelve a arrancar quagga (`/etc/init.d/quagga start`)
- En algunos escenarios puede que algunos *routers* estén preconfigurados para que arranquen con quagga ya lanzado.

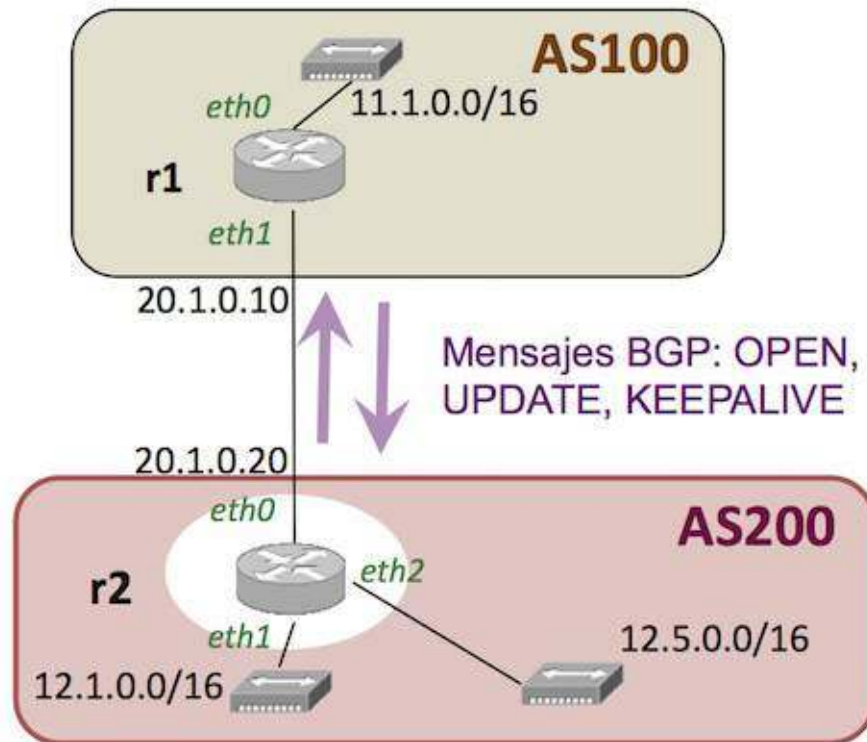
Monitorización de la configuración: `vttysh`

```
r1:~# vtysh
Copyright 1996-2005 Kunihiro Ishiguro, et al.

r1# ?
clear          Reset functions
configure     Configuration from vty interface
copy          Copy from one file to another
debug         Debugging functions (see also 'undebug')
disable       Turn off privileged mode command
end           End current mode and change to enable mode
exit          Exit current mde an down to previous mode
list          Print command list
no            Negate a command or set its defaults
ping          Send echo messages
quit          Exit current mode and down to previous mode
show          Show running system information
ssh           Open an ssh connection
start-shell   Start UNIX shell
telnet        Open a telnet connection
terminal      Set terminal line parameters
traceroute    Trace route to destination
undebug       Disable debugging functions (see also 'debug')
write         Write running configuration to memory, network, or terminal
r1#
```

Tabla de encaminamiento BGP

El comando `show ip bgp` muestra la información sobre la tabla de encaminamiento BGP del *router* (el ejemplo muestra la configuración del *router* `r2` de la figura):



```
r2# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>11.1.0.0/16	20.1.0.10	0			100 i
*>20.1.0.0/16	0.0.0.0	0		32768	i
*>12.1.0.0/16	0.0.0.0	0		32768	i
*>12.5.0.0/16	0.0.0.0	0		32768	i

Donde `*` indica que la ruta es válida y `>` indica que la ruta ha sido elegida como ruta preferida.

Observando la primera fila de la tabla:

Network	Next Hop	Metric	LocPrf	Weight	Path
*>11.1.0.0/16	20.1.0.10	0			100 i

Desde `r2` se alcanza la subred 11.1.0.0/16 a través de 20.1.0.10, utilizando un atributo `AS_PATH=100` ya que sólo se atraviesa dicho sistema autónomo para llegar a esa subred. El último parámetro es el atributo `ORIGIN`, en este caso `i` que indica si la subred ha sido aprendida mediante:

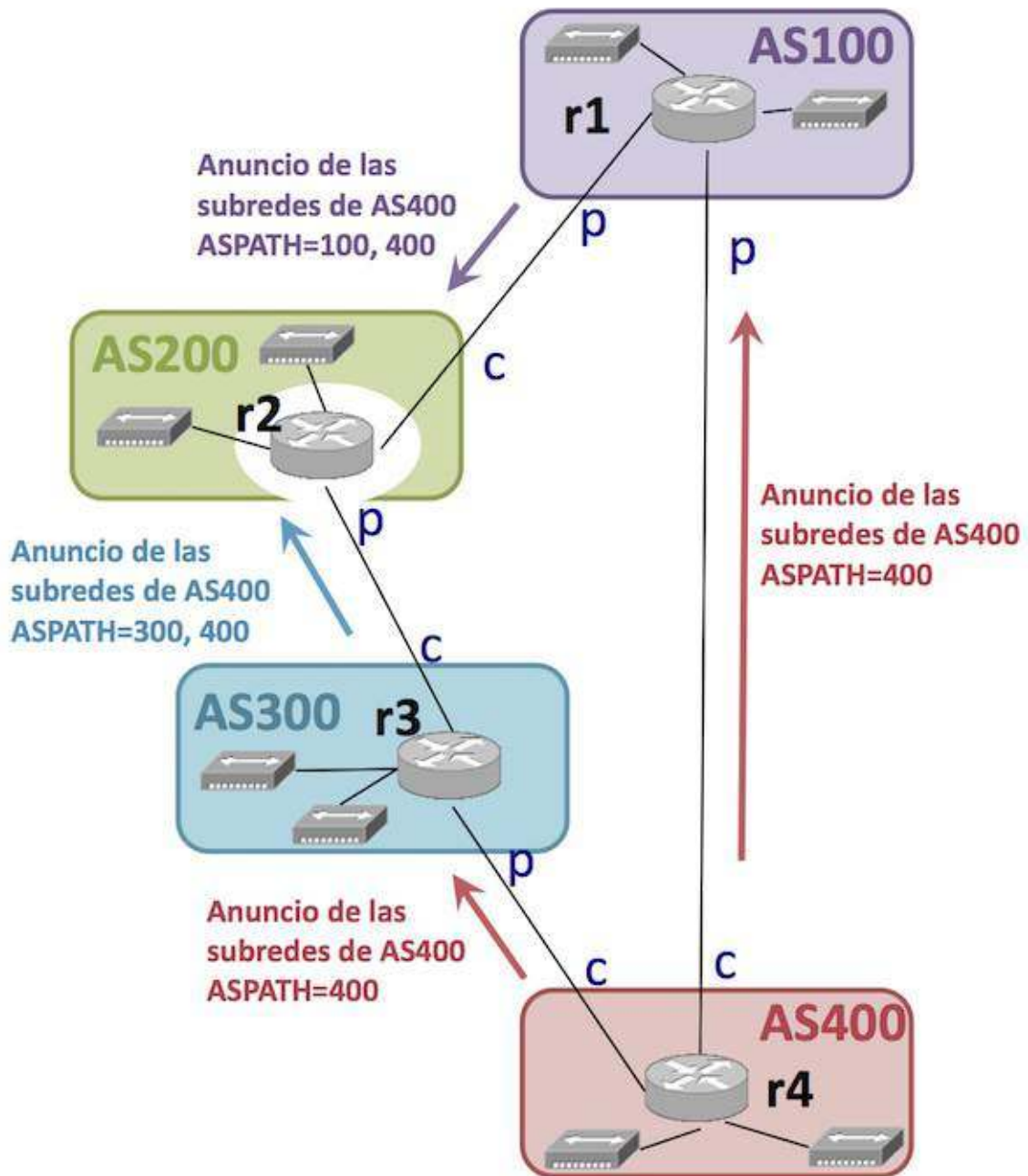
- `i`: líneas network, agregación de direcciones rutas por defecto en `bgpd.conf`
- `e`: otro protocolo externo

- ? : líneas redistribute en bgpd.conf

En la tabla BGP se muestra el valor AS_PATH asociado a cada ruta.

Cada AS añade su número de sistema autónomo al atributo AS_PATH antes de anunciar una ruta. Para las subredes de AS400, 16.0.0.0/16:

- AS300 añade el identificador 300 antes de exportar dichas subredes a AS200. Por tanto, AS200 recibe de AS300: AS_PATH=300 400.
- AS100 añade el identificador 100 antes de exportar dichas subredes a AS200. Por tanto, AS200 recibe de AS100: AS_PATH=100 400.



```
r2# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>16.0.0.0/16	20.2.0.30			0	300 400 i
*	20.1.0.10			0	100 400 i

El AS_PATH puede observarse en la columna `Path` .

Redistribución de rutas entre BGP y OSPF

Un router puede ejecutar varios protocolos de encaminamiento diferentes. Así, por ejemplo, un router frontera de un AS ejecutará tanto BGP como un protocolo interior (RIP u OSPF).

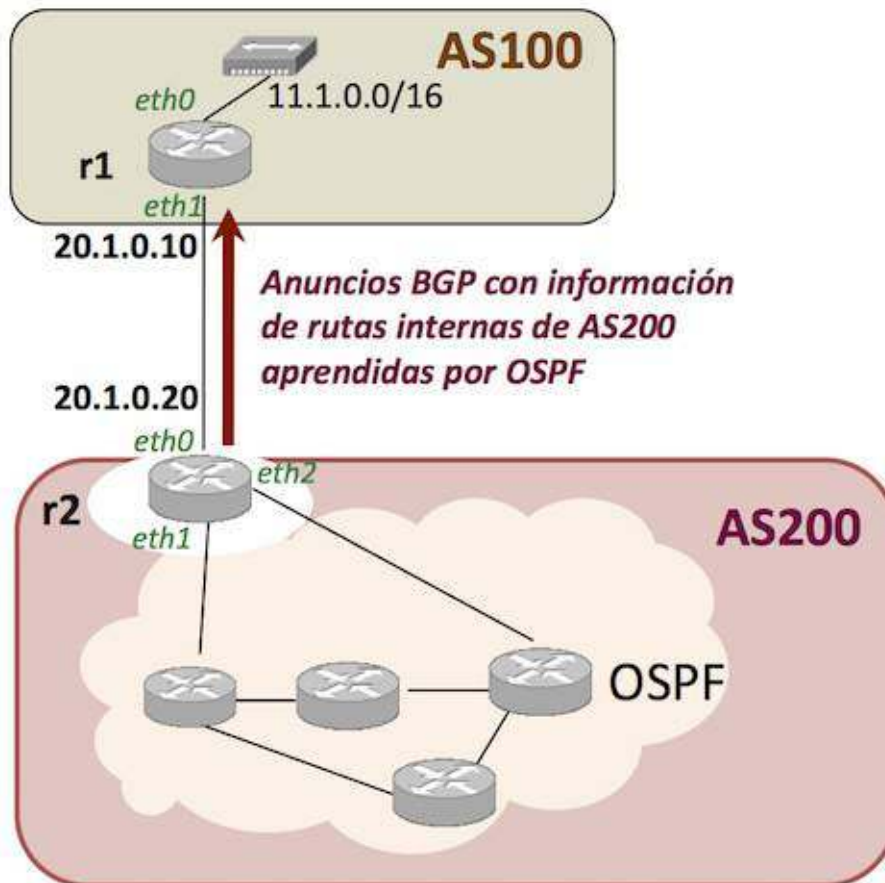
Para que las rutas aprendidas por OSPF se propaguen hacia el exterior anunciándose a través de BGP es necesario configurarlo explícitamente en el fichero `bgpd.conf` .

Para que las rutas aprendidas por BGP se propaguen internamente utilizando OSPF es necesario configurarlo explícitamente en el fichero `ospfd.conf` .

Rutas aprendidas por OSPF en anuncios BGP

Si AS200 está ejecutando OSPF entre todos sus routers internos, el router frontera `r2` debe estar ejecutando tanto BGP como OSPF. El fichero `daemons` tendrá activado: `zebra` , `ospf` y `bgpd` .

En el siguiente ejemplo se muestra la configuración de `r2` :



- Los ficheros `ospfd.conf` de cada uno de los routers de AS200 estarán configurados adecuadamente para que se anuncien por OSPF las rutas interiores de AS200.
- Para que las rutas aprendidas por `r2` a través de OSPF se anuncien por BGP es necesario añadir la siguiente línea en el fichero `bgpd.conf` :

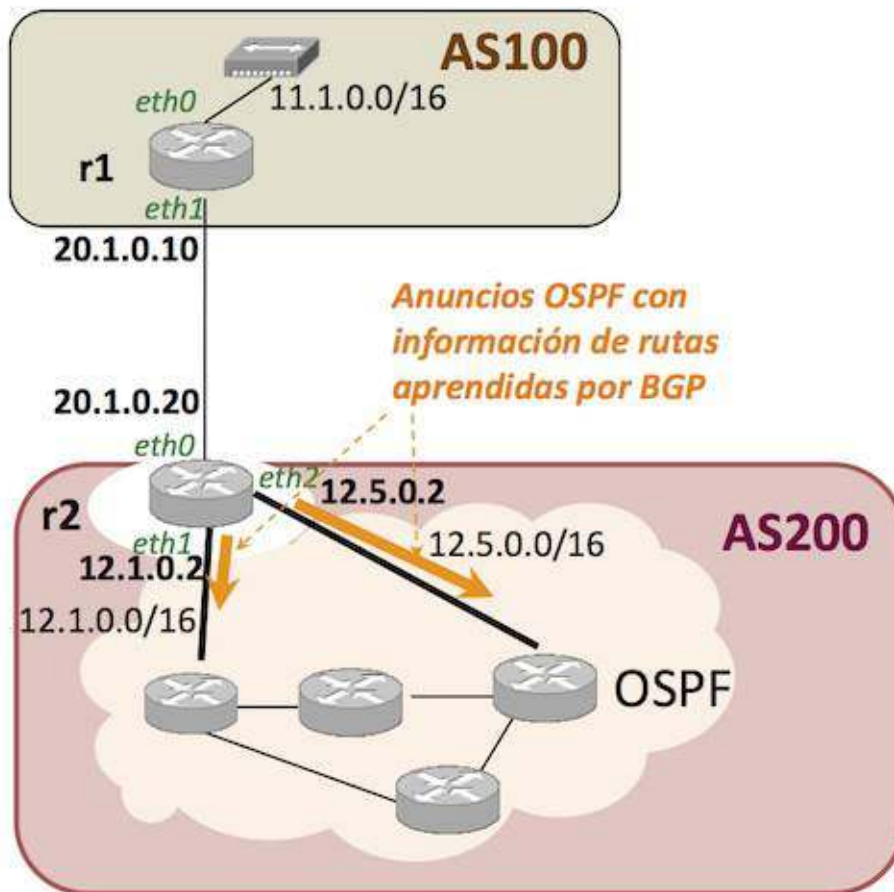
```

...
router bgp 200
...
  redistribute ospf
...

```

Rutas aprendidas por BGP en anuncios OSPF

En el siguiente ejemplo se muestra la configuración de `r2` :



- Para que los routers internos de AS200 puedan alcanzar los destinos de AS100, `r2` puede redistribuir la información que ha aprendido por BGP utilizando OSPF. Para ello, el fichero `ospfd.conf` de `r2` debe incluir la siguiente línea:

```
...
router ospf
...
  redistribute bgp
...
```

`r2` NO tiene en su fichero de configuración `ospfd.conf` la línea `network 20.1.0.0/16`, ya que en esa subred no hay otros routers OSPF.

Para que se anuncie la subred 20.1.0.0/16 dentro del AS es necesario añadir en

`ospfd.conf` la línea:

```
redistribute connected
```

Así, en `r2` el fichero `ospfd.conf` quedaría:

```

...
router ospf
  network 12.1.0.0/16 area 0
  network 12.5.0.0/16 area 0
  redistribute connected
  redistribute bgp
...

```

Políticas de exportación de rutas

- Las relaciones entre ASs dictan unas reglas de exportación de rutas.
- Para especificar estas políticas, hay que incluir la relación de rutas que se exportan en el fichero bgpd.conf.
- Para cada vecino es necesario decidir si se exporta o no una determinada información:

- Para un vecino BGP `<vecino-BGP>` se define un filtro **out** (de exportación) al cuál le asignamos un nombre `<nombreLista>` :

```
neighbor <vecino-BGP> filter-list <nombreLista> out
```

- El filtro se define como una lista de reglas expresadas como sentencias `deny/permit` de patrones de `AS_PATH`.

```

ip as-path access-list <nombreLista> [deny/permit] <patrónAS_PATH>
...
ip as-path access-list <nombreLista> [deny/permit] <patrónAS_PATH>

```

Las reglas de un filtro se aplicarán en el orden en el que las hemos escrito.

- Para cada subred a exportar (ruta preferida) a dicho vecino BGP, se tomará su atributo `AS_PATH` que aparece en la tabla BGP del router, si se cumple el patrón de la regla del filtro `<patrónAS_PATH>` , se aplicará la acción (`deny/permit`) y por tanto se enviará o no el anuncio.
- Si no se cumple ninguno de los patrones `AS_PATH` de las sentencias del filtro, la acción por defecto es `deny` . No se anuncia.

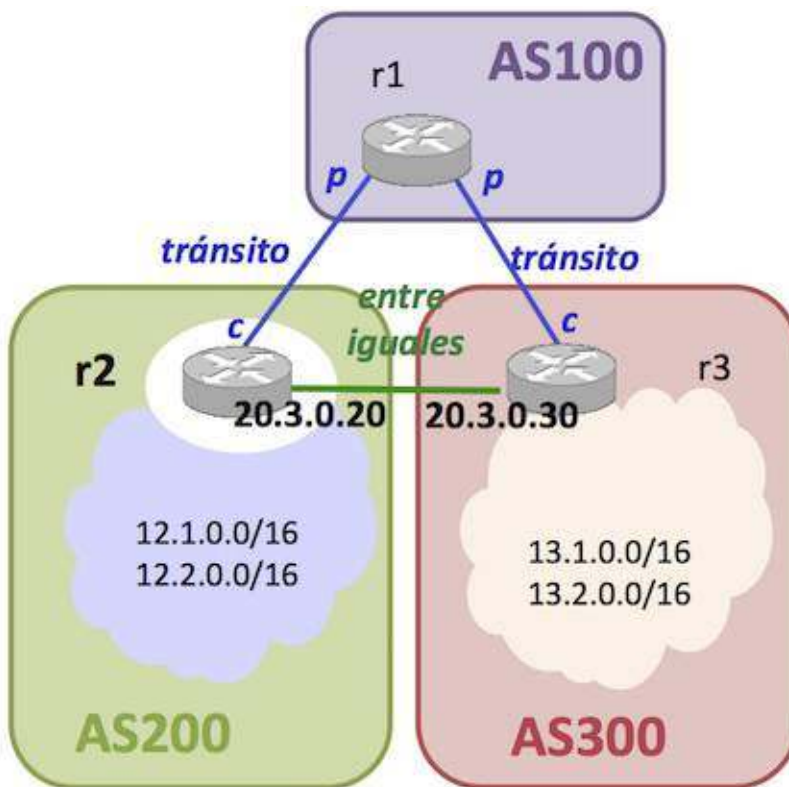
Patrón para `AS_PATH` en las políticas de exportación

- Algunos ejemplos de `<patrónAS_PATH>` para utilizar en las políticas de exportación:

Patrón	Significado
<code>^100</code>	cualquier AS_PATH que comience con AS100. Cualquier ruta que me haya anunciado AS100.
<code>100\$</code>	cualquier AS_PATH que termine con AS100. Cualquier ruta que haya sido generada en AS100.
<code>^\$</code>	AS_PATH vacío. Cualquier ruta originada en mi propio AS.
<code>^100_200\$</code>	el AS_PATH dado por 100,200.
<code>.*</code>	Cualquier AS_PATH.

Ejemplo de políticas de exportación de rutas

Configuración de `bgpd.conf` en `r2` :



```
router bgp 200
  neighbor 20.3.0.30 remote-as 300

  neighbor 20.3.0.30 filter-list listaHaciaAS300 out

  redistribute ...
  redistribute ...

  aggregate-address ...
  aggregate-address ...

  ip as-path access-list listaHaciaAS300 deny ^100
  ip as-path access-list listaHaciaAS300 permit .*
```

- Desde `r2` no se envía ningún anuncio a `r3` que contenga subredes con `AS_PATH` cuyo primer AS sea AS100. Por tanto, no se envían las subredes que me haya anunciado AS100. Sí se permite el envío del resto de las subredes.

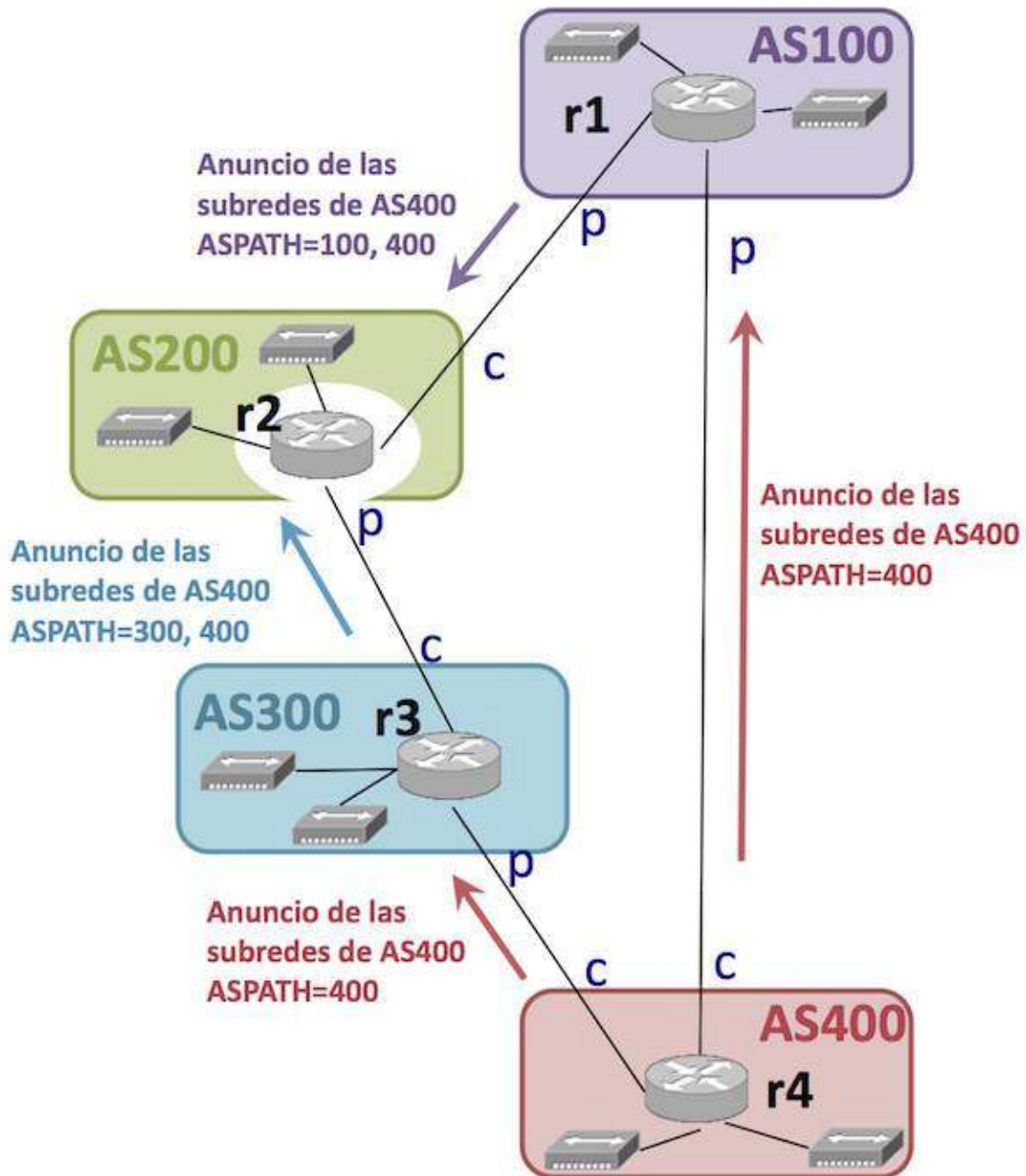
Selección de la mejor ruta

- Por defecto, todas las subredes recibidas en anuncios BGP se incorporan a la tabla BGP (BGP RIB). Por tanto, en esta tabla se muestran todas las posibilidades que conoce un router para alcanzar cada una de las subredes.
- Si existe más de una posibilidad de ruta para alcanzar una determinada subred, BGP seleccionará cuál será la ruta preferida atendiendo a los siguientes criterios (sólo los criterios resaltados en negrita son los que tendremos en cuenta al realizar las prácticas):
 1. Si el siguiente salto que debe utilizarse para alcanzar una subred es inaccesible, se descarta la ruta.
 2. Se prefiere ruta con mayor valor de Weight (parámetro configurable en Cisco y quagga).
 3. **Se prefiere ruta con mayor valor de atributo `LOCAL_PREF`.**
 4. Se prefiere ruta generada localmente en el fichero de configuración de BGP.
 5. **Se prefiere ruta con el atributo `AS_PATH` más corto.**
 6. Se prefiere ruta en función de atributo ORIGIN: IGP mejor que EGP y mejor que INCOMPLETE.
 7. Se prefiere ruta con menor atributo Multi-Exit Discriminator (MED).
 8. ...
- Una vez seleccionadas las rutas preferidas:
 - Las rutas preferidas se incorporan a la tabla de encaminamiento de la máquina.

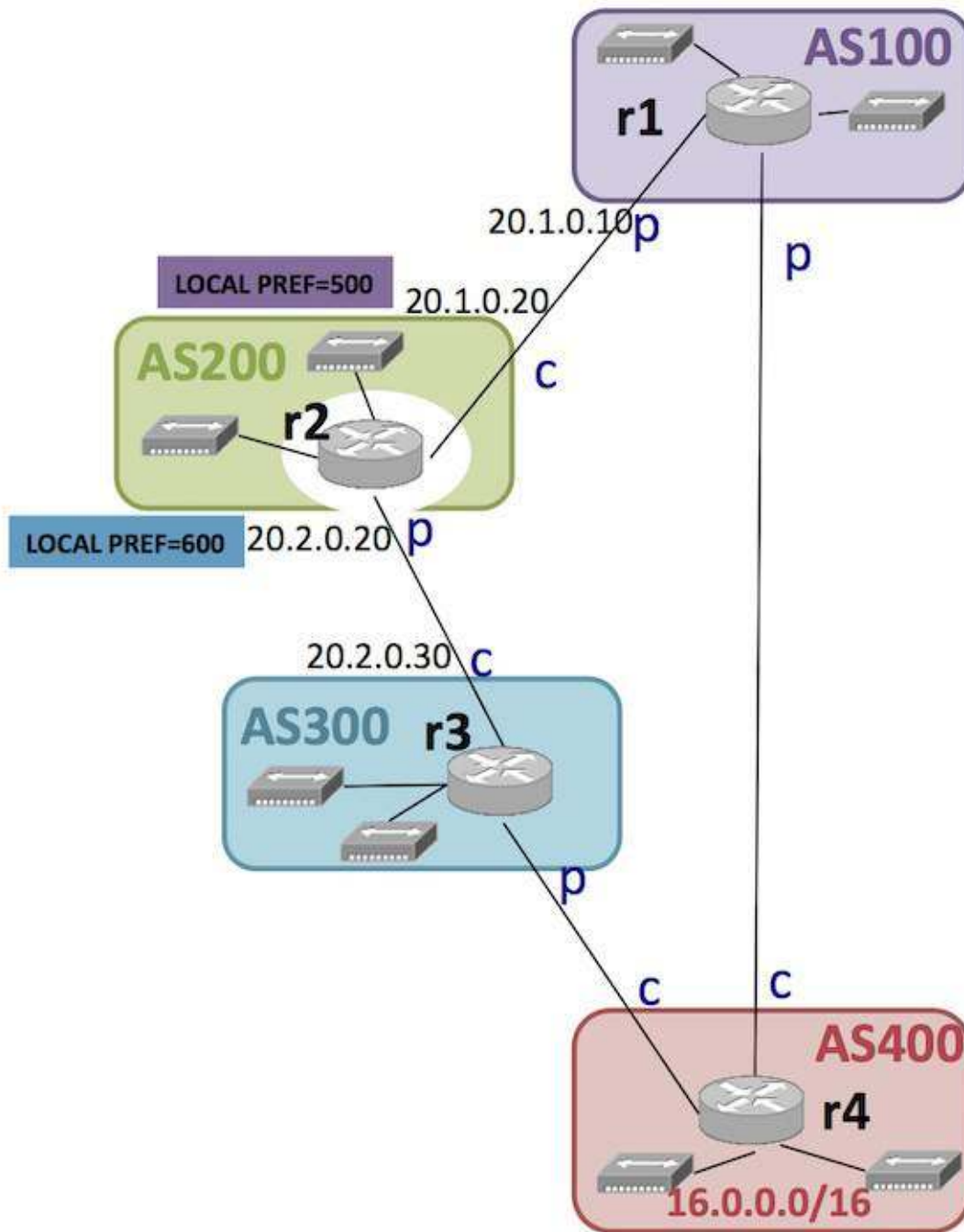
- Las rutas preferidas se anuncian (si es conveniente) a los ASs vecinos.

Configuración del atributo LOCAL PREF

- Cuando un *router* recibe diferentes rutas para alcanzar un mismo destino, incluye todas ellas en su tabla BGP y decide cuál es la mejor ruta en función de un LOCAL PREF mayor. En caso de empate, se elegirá por el ASPATH más corto (existen más criterios para la selección pero no los estudiaremos).
- LOCAL PREF es un atributo que sólo tiene sentido dentro de un AS y no se propaga fuera del mismo.
- En la figura, `r2` debería elegir la ruta hacia AS400 a través de AS300 (su cliente). Sin embargo en la configuración BGP de `r2` no se puede expresar si un vecino BGP es proveedor, cliente o mantiene con él una relación entre iguales.



- Para que r2 seleccione la ruta hacia AS400 a través de AS300, en r2 el atributo LOCAL PEF del vecino de AS300 debe ser mayor que el LOCAL PEF del vecino de AS100.



```

router bgp 200
  neighbor 20.1.0.10 remote-as 100
  neighbor 20.2.0.30 remote-as 300

  neighbor 20.1.0.10 route-map confLocalPrefAS100 in
  neighbor 20.2.0.30 route-map confLocalPrefAS300 in

  redistribute ...
  redistribute ...

  aggregate-address ...
  aggregate-address ...

  route-map confLocalPrefAS100 permit 10
    set local-preference 500

  route-map confLocalPrefAS300 permit 10
    set local-preference 600

```

- En la tabla BGP de `r2` se muestra el valor LOCAL PREF asociado a cada ruta.
- En `r2` se puede consultar la tabla BGP para ver los atributos LOCAL PREF asignados a las rutas aprendidas.

```

r2# show ip bgp
BGP table version is 0, local router ID is 10.1.0.20
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

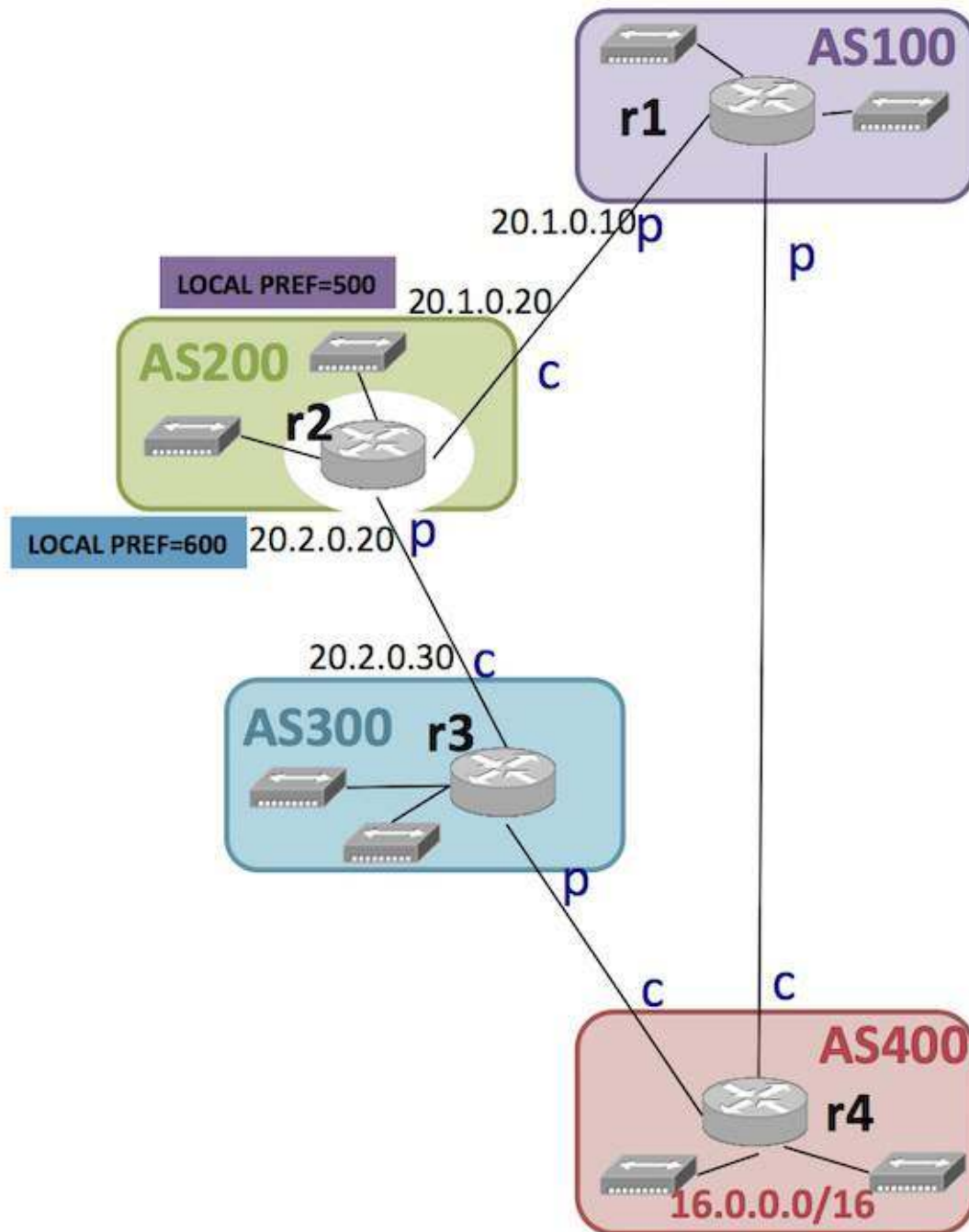
```

Network	Next Hop	Metric	LocPrf	Weight	Path
*>16.0.0.0/16	20.2.0.30		600	0	300 400 i
*	20.1.0.10		500	0	100 400 i

- Si en la tabla BGP no aparece valor LOCAL PREF para una ruta, éste parámetro tomará su valor por defecto que es 100.

Configuración de una ruta por defecto

- Si un AS tiene un solo proveedor, su proveedor podría anunciarle simplemente una ruta por defecto, en vez de todas las subredes que conoce.
- Así, por ejemplo, r2 podría anunciar a r3 todas las subredes que conoce con una ruta por defecto



- Para configurarlo, en el fichero de configuración `bgpd.conf` de `r2` :

```
router bgp 200
...
neighbor 20.2.0.30 remote-as 300
neighbor 20.2.0.30 default-originate
...
```

- La línea `default-originate` por sí sola no evita los anuncios de las redes originales, sino que simplemente anuncia una ruta por defecto.

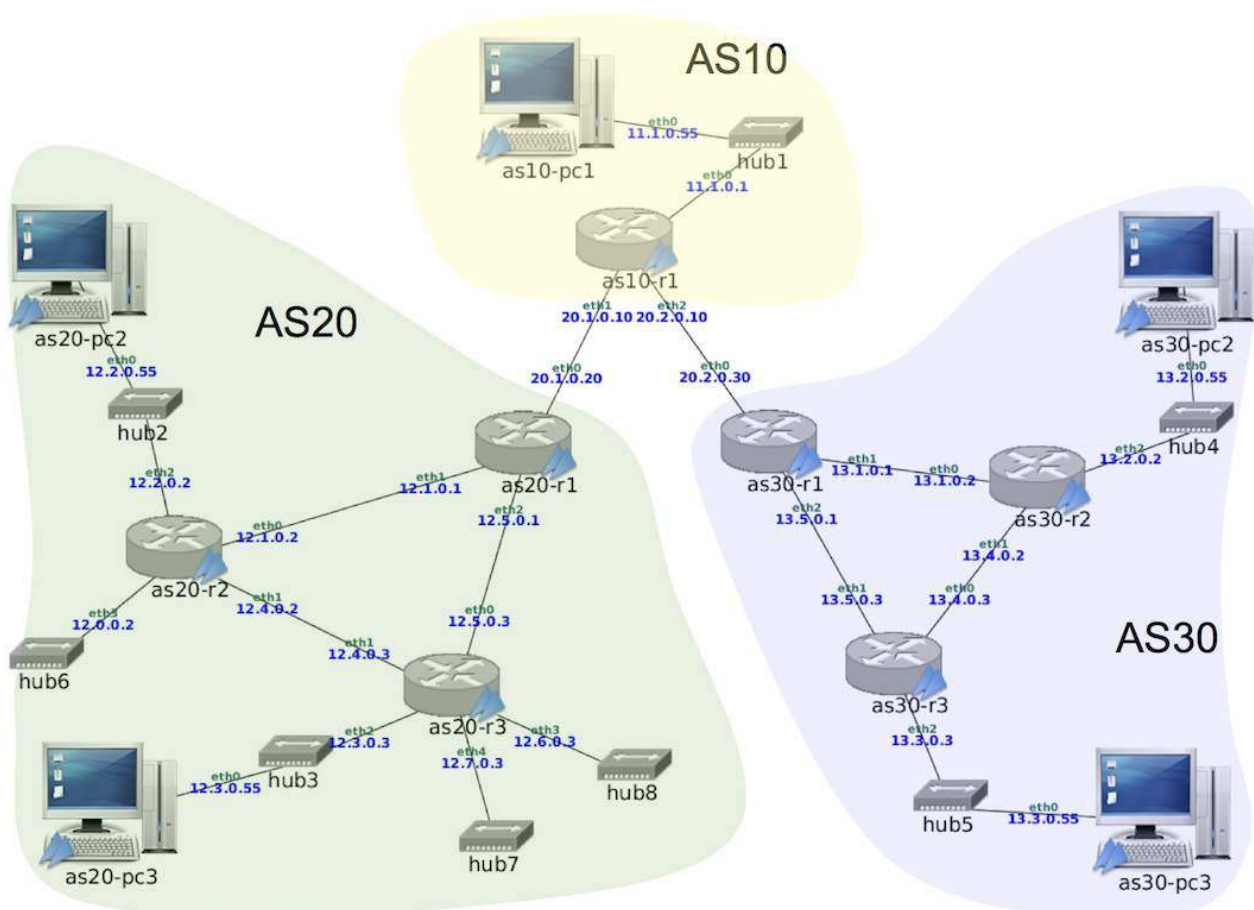
- Para eliminar los anuncios de las redes originales hay que crear adicionalmente para ese vecino una `filter-list` que evite esos anuncios:

```
router bgp 200
...
neighbor 20.2.0.30 remote-as 300
neighbor 20.2.0.30 default-originate
neighbor 20.2.0.30 filter-list listaHaciaAS300 out
...
redistribute ...
...
aggregate-address ...
...
ip as-path access-list listaHaciaAS300 deny .*
...
```

Laboratorio de BGP

- Relaciones de tránsito
 - pcs y routers de AS20 con OSPF
 - pc y router de AS10
 - pcs y routers de AS30 con OSPF
 - Agregación de rutas
- Relación entre iguales
- Políticas de exportación y orden de preferencia en la selección de rutas
 - Ruta por defecto

En el fichero [lab-BGP.tgz](#) se encuentran los ficheros de configuración para crear un escenario de red como el que se muestra en la figura.



En esta figura hay tres AS (Sistemas Autónomos): AS10, AS20 y AS30. Se ha configurado OSPF dentro de AS20 y AS30 y se desea configurar BGP para conectar los 3 sistemas autónomos. En AS10 no se utilizará ningún protocolo de encaminamiento interior ya que sólo hay una red interna.

Se realizará la configuración por pasos. Supondremos para la configuración de BGP que AS10 y AS20 mantienen una relación de tránsito y que AS10 y AS30 mantienen una relación de tránsito, siendo en ambos casos AS10 el proveedor.

Relaciones de tránsito

pcs y routers de AS20 con OSPF

1. Descomprime el fichero de configuración del escenario `lab-BGP.tgz` . Al arrancar NetGUI debes abrir el escenario definido en el directorio `lab-BGP` .
2. Arranca de una en una sólo las máquinas de AS20.
3. Las máquinas tienen configurada una dirección IP en cada una de sus interfaces de red. Los pcs además tienen configurada una ruta por defecto al único *router* al que están directamente conectados.
4. Los *routers* de AS20 (`as20-r1` , `as20-r2` y `as20-r3`) tienen configurado el protocolo OSPF para que intercambien información de encaminamiento dentro de AS20 (consulta los ficheros `daemons` y `ospfd.conf` de estos *routers*). Arranca `quagga` en todos los *routers* de AS20.
5. Consulta las tablas de encaminamiento utilizando la interfaz VTY con los procesos `ospfd` de cada *router* y mediante el comando `route` .
6. Comprueba utilizando `ping` que todos los pcs y *routers* tienen conectividad dentro de AS20.
7. Modifica la configuración de `quagga` en `as20-r1` para que utilice además de OSPF el protocolo BGP. Define como vecino suyo a `as10-r1` . Utiliza únicamente la redistribución de las subredes directamente conectadas (`redistribute connected`). No uses aún la redistribución de rutas entre OSPF y BGP. Reinicia `quagga` en `as20-r1` .
8. ¿Debería haber aprendido alguna ruta `as20-r1` por BGP? Compruébalo consultando la tabla de encaminamiento mediante el mandato `route` y conectándote a la interfaz VTY del proceso `bgpd` .
9. ¿Deberían haber aprendido alguna ruta `as20-r2` y `as20-r3` ?

pc y router de AS10

1. Arranca `as10-pc1` y `as10-r1`. La máquina `as10-pc1` tiene configurada una dirección IP y una ruta por defecto al *router* al que está directamente conectado. El *router* `as10-r1` tiene configuradas direcciones IP, una en cada interfaz, pero no tiene configurada ninguna ruta adicional a las de las subredes a las que está directamente conectado.
2. Configura `quagga` en `as10-r1` para que utilice el protocolo BGP. Define como vecinos suyos a `as20-r1` y a `as30-r1`.
3. Captura el tráfico con `tcpdump` en `as20-r1(eth0)`, con la opción `-s` para capturar paquetes enteros y `-w` para guardar la captura en un fichero.
4. Arranca `quagga` en `as10-r1`. Espera un minuto e interrumpe la captura.
5. Analiza la captura realizada:
 - Observa que el tráfico de BGP va dentro de una conexión TCP.
 - Localiza los mensajes `OPEN` que intercambian los *routers* vecinos. Observa en ambos mensajes `OPEN` los siguientes campos:
 - `My AS`
 - Identificador del *router* BGP
 - `Hold time`
 - En los parámetros opcionales, el campo `Capabilities Advertisement` que contiene información del número de sistema autónomo de 32 bits.
 - Localiza los mensajes `KEEPALIVE` que intercambian los *routers*. Además de la cabecera obligatoria de BGP (`Marker`, `Length` y `Type`) ¿qué otra información viaja en este tipo de mensajes?
 - Localiza los mensajes `UPDATE` que intercambian los *routers*. Observa en ambos mensajes `UPDATE` los siguientes campos:
 - Rutas eliminadas
 - Rutas anunciadas
 - Atributos, en particular el valor de `NEXT_HOP` y `AS_PATH`.
 - El atributo `ORIGIN` tiene como valor `INCOMPLETE` porque esas subredes se anuncian debido a la redistribución de subredes y no a través de líneas `network`. Elimina en el fichero `bgpd.conf` de `as10-r1` la línea:

```
redistribute connected
```

y añade las siguientes líneas:

```
network 11.1.0.0/16
network 20.1.0.0/16
network 20.2.0.0/16
```

Interrumpe `quagga` en `as10-r1`, inicia una captura de tráfico en `as20-r1(eth0)` y arranca `quagga` en `as10-r1` de nuevo. Pasado 2 minutos aproximadamente, fíjate en el valor del atributo `ORIGIN` en el mensaje `UPDATE` que genera `as10-r1` para dichas subredes. Puedes observar como el valor de este atributo (i=IGP, e=EGP, ?=incomplete) también se observa en la tabla BGP, en la columna `Path`, junto al ASN que originó el anuncio.

6. Consulta la tabla de encaminamiento utilizando la interfaz VTY con el proceso `bgpd` y con el comando `route` en los *routers* `as10-r1` y `as20-r1`. Observa las diferencias.
7. Prueba a hacer un `ping` desde `as10-pc1` hacia `as20-pc2` y comprueba que no funciona. ¿Por qué? (Explica la tabla de encaminamiento que tiene `as10-r1`).
8. Modifica la configuración del fichero `bgpd.conf` de `as20-r1` para que se redistribuyan las rutas aprendidas por OSPF de AS20 a otros ASs utilizando BGP. Reinicia `quagga` en `as20-r1`. Comprueba que ahora `as10-r1` tiene rutas a todas las redes de AS20. En la tabla BGP de `as10-r1` fíjate en el atributo `ORIGIN` para las subredes de AS20.
9. Prueba a hacer un `ping` desde `as10-pc1` hacia `as20-pc2` y comprueba que todavía no funciona. ¿Por qué? (Explica la tabla de encaminamiento que tiene `as20-r2`).
10. Modifica la configuración del fichero `ospfd.conf` de `as20-r1` para que se redistribuyan las rutas aprendidas por BGP a los *routers* de AS20 mediante OSPF. Reinicia `quagga` en `as20-r1`. Comprueba que `as20-r2` tiene ruta a la red de AS10.
11. Comprueba que ahora sí funciona el `ping` entre `as10-pc1` y `as20-pc2`.

pcs y routers de AS30 con OSPF

1. Arranca todas las máquinas de AS30 de una en una. Los *routers* tienen configurada una dirección IP por cada una de sus interfaces. Los pcs tienen configurada una dirección IP y una ruta por defecto al *router* al que están directamente conectados.
2. Los *routers* de AS30 (`as30-r1`, `as30-r2` y `as30-r3`) tienen configurado el protocolo OSPF para que intercambien información de encaminamiento dentro de AS30 (consulta los ficheros `daemons` y `ospfd.conf` de estos *routers*). Arranca `quagga` en todos los *routers* de AS30.

3. Consulta las tablas de encaminamiento utilizando la interfaz VTY con los procesos `ospfd` de cada *router* y mediante el comando `route` .
4. Comprueba utilizando `ping` que todos los pcs y *routers* tienen conectividad dentro de AS30.
5. Modifica la configuración de `quagga` en `as30-r1` para que utilice además el protocolo BGP. Define como vecino suyo a `as10-r1` . No uses aún la redistribución de rutas entre OSPF y BGP.
6. Captura el tráfico con `tcpdump` en `as10-r1(eth2)` , con la opción `-s` para capturar paquetes enteros y `-w` para guardar la captura en un fichero.
7. Reinicia `quagga` en `as30-r1` . Espera un minuto e interrumpe la captura.
8. Analiza la captura realizada:
 - Observa que el tráfico de BGP va dentro de una conexión TCP.
 - Localiza los mensajes `OPEN` que intercambian los *routers* vecinos. Observa en ambos mensajes `OPEN` los siguientes campos:
 - `My AS`
 - Identificador del *router* BGP
 - `Hold time`
 - En los parámetros opcionales, el campo `Capabilities Advertisement` que contiene información del número de sistema autónomo de 32 bits.
 - Localiza los mensajes `KEEPALIVE` que intercambian los *routers*. Comprueba que son similares a los que ya observaste en el apartado anterior
 - Trata de suponer qué rutas le anunciará `as10-r1` a `as30-r1` en sus mensajes `UPDATE` . ¿Qué `AS_PATH` crees que traerán esas rutas? Como los atributos de un mensaje `UPDATE` son comunes a todas las rutas anunciadas, ¿podrá anunciar `as10-r1` todas las subredes que conoce en un solo mensaje `UPDATE` ?
 - Trata de suponer qué rutas le anunciará `as30-r1` a `as10-r1` en sus mensajes `UPDATE` .
 - Localiza en la captura los mensajes `UPDATE` que intercambian los *routers* y confirma si tus suposiciones son ciertas. Observa el valor de los atributos `NEXT_HOP` y `AS_PATH` .
9. ¿Debería haber aprendido alguna ruta `as30-r1` ? Compruébalo consultando la tabla de encaminamiento mediante el mandato `route` .

10. El resto de *routers* de AS30 ¿deberían haber aprendido alguna otra ruta? Compruébalo.
11. Prueba a hacer un `ping` desde `as10-pc1` hacia `as30-pc3` y comprueba que no funciona. ¿Por qué? (Explica la tabla de encaminamiento de `as10-r1`).
12. Modifica la configuración del fichero `bgpd.conf` de `as30-r1` para que se redistribuyan las rutas aprendidas por OSPF de AS30 a otros ASs utilizando BGP. Reinicia `quagga` en `as30-r1` . Comprueba que ahora `as10-r1` tiene rutas a todas las redes de AS30. En la tabla BGP de `as10-r1` fíjate en el atributo `ORIGIN` para las subredes de AS30.
13. Prueba a hacer un `ping` desde `as10-pc1` hacia `as30-pc3` y comprueba que todavía no funciona. ¿Por qué? (Explica la tabla de encaminamiento de `as30-r3`).
14. Modifica la configuración del fichero `ospfd.conf` de `as30-r1` para que se redistribuyan las rutas aprendidas por BGP a los *routers* de AS30 mediante OSPF. Reinicia `quagga` en `as30-r1` . Comprueba que ahora `as30-r3` tiene ruta a la red de AS10.
15. Comprueba que ahora sí funciona el `ping` entre `as10-pc1` y `as30-pc3` .
16. Comprueba que hay conectividad entre todos los pcs de la figura.

Agregación de rutas

La configuración de BGP realizada en el apartado anterior provoca que las subredes del sistema autónomo AS20 se almacenen de forma independiente, ocupando cada una de ellas una entrada diferente en las tablas de encaminamiento de los *routers* de AS10 y AS30. De forma equivalente, cada una de las subredes de AS30 ocupan entradas diferentes en las tablas de encaminamiento de los *routers* de AS10 y AS20.

Utilizando CIDR pueden agruparse estas entradas para que los anuncios por BGP que emiten AS20 y AS30 optimicen el número de entradas en las tablas de encaminamiento en los *routers* externos a dichos sistemas autónomos.

1. Interrumpe la ejecución de `quagga` en `as20-r1` y `as30-r1` .
2. Configura BGP en AS20 para que se optimice el número de entradas en las tablas de encaminamiento de los *routers* de AS10 y AS30. Ten en cuenta que al realizar la agregación de rutas, dicha agregación sólo puede referirse a subredes que pertenezcan a AS20.
3. Configura BGP en AS30 para que se optimice el número de entradas en las tablas de encaminamiento de los *routers* de AS10 y AS20. Ten en cuenta que al realizar la agregación de rutas, dicha agregación sólo puede referirse a subredes que

pertenezcan a AS30.

4. Captura el tráfico con `tcpdump` en `as10-r1(eth2)` , con la opción `-s` para capturar paquetes enteros y `-w` para guardar la captura en un fichero.
5. Inicia `quagga` en `as20-r1` y `as30-r1` .
6. Analiza la captura realizada:
 - Trata de suponer cómo serán los nuevos mensajes `UPDATE` que intercambien los *routers* anunciando las redes de AS20 y AS30. Localízalos en la captura y confirma si tus suposiciones son ciertas.
 - Fíjate en el atributo `ORIGIN` para estas subredes en los mensajes `UPDATE` y en la tabla BGP de `as10-r1` , su valor es diferente después de realizar la agregación.
7. Consulta las tablas de encaminamiento de los *routers* de AS20 y AS30 mediante el comando `route` , para ver cómo se han agregado las rutas hacia el sistema autónomo externo.
8. Consulta la tabla BGP en `as20-r1` , observarás como las subredes de AS20 con el prefijo agregado aparecen como ruta preferida y serán las que se anuncian a otros vecinos BGP. Además, las subredes que se anunciaban previamente de forma independiente y ahora se anuncian dentro del prefijo agregado también aparecen pero marcadas con una `s` que indica que se suprimen. Consulta esta misma información en la tabla BGP de `as30-r1` para las subredes de AS30 que agrega este router.

Relación entre iguales

AS20 y AS30 se dan cuenta de que intercambian mucho tráfico entre ellos y deciden conectar directamente `as20-r1` y `as30-r1` definiendo una relación entre iguales entre los mismos.

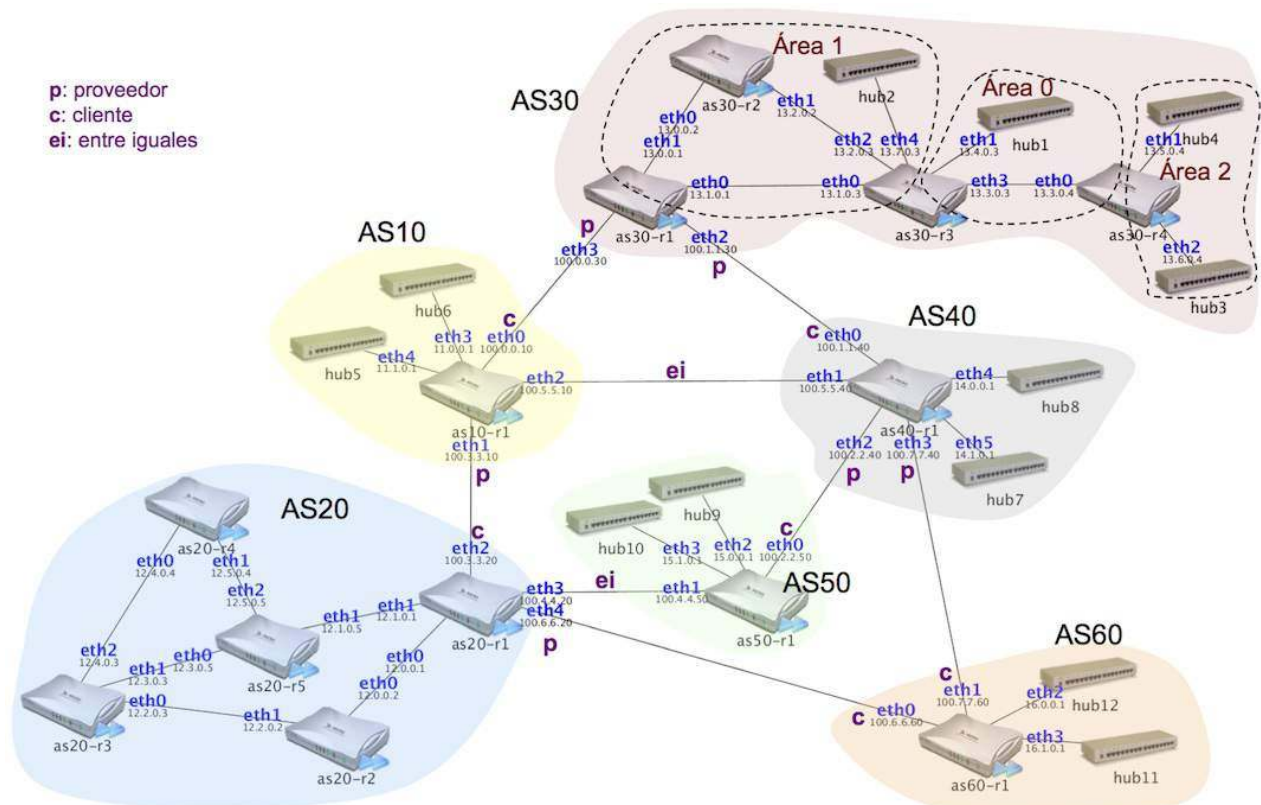
1. Interrumpe `quagga` en `as20-r1` y `as30-r1` .
2. Interrumpe la ejecución de los *routers* `as20-r1` y `as30-r1` , apagando cada uno de ellos desde la interfaz gráfica de NetGUI. Dibuja un enlace directo entre ambos *routers* y arráncalos de nuevo, uno después de otro.
3. Asigna la dirección 20.3.0.20 a la nueva interfaz de `as20-r1` y la dirección 20.3.0.30 a la nueva interfaz de `as30-r1` , ambas direcciones de la red 20.3.0.0/16.

4. No apliques por ahora ninguna política de exportación de rutas. Modifica la configuración de BGP de `as20-r1` y `as30-r1` para añadir en cada uno al otro como nuevo vecino. Por defecto si no se configuran políticas de exportación, se anuncian todas las rutas seleccionadas como preferidas.
5. Arranca `quagga` en `as20-r1` y `as30-r1`.
6. Comprueba mediante `route` en `as20-r1` la ruta hacia las redes de AS30, y en `as30-r1` la ruta hacia las redes de AS20. Utilizando la interfaz VTY en ambos *routers* observa cómo cada uno tiene dos rutas alternativas para el sistema autónomo vecino, y ha elegido una de ellas. ¿Cuál? ¿Por qué? Ten en cuenta que `LOCAL_PREF` no se ha modificado y por tanto valdrá para todas las interfaces su valor por defecto, 100.
7. Observa la tabla BGP de `as20-r1` e indica cuántas rutas alternativas existen para las subredes 20.1.0.0/16, 20.2.0.0/16 y 20.3.0.0/16. Indica cómo `as20-r1` ha aprendido estas rutas alternativas y cuál se ha seleccionado como preferida.
8. ¿Qué ruta crees que seguirán los paquetes intercambiados entre `as20-pc3` y `as30-pc2`? Compruébalo.
9. ¿Qué ruta crees que seguirán los paquetes enviados desde `as30-pc3` con destino `as10-pc1`? Compruébalo utilizando `traceroute`. Utilizando la interfaz VTY en `as30-r1` comprueba cómo tiene dos rutas alternativas para la red `11.1.0.0/16`. Observa cuál es la elegida y por qué.
10. Apaga la interfaz `eth0` de `as30-r1` con `ifconfig eth0 down`. Espera unos 3 minutos. ¿Qué habrá pasado ahora con la ruta que seguirán los paquetes enviados desde `as30-pc3` con destino `as10-pc1`? Compruébalo utilizando `traceroute`. Utilizando la interfaz VTY en `as30-r1` comprueba que ahora sólo tiene una ruta para la red `11.1.0.0/16`. Dada las relaciones entre AS10, AS20 y AS30 indica si esta situación perjudica a alguno de los AS y por qué.
11. Teniendo en cuenta las relaciones entre AS10, AS20 y AS30:
 - i. ¿Qué rutas debería exportar AS20 a AS30, y qué rutas no debería exportarle?
 - ii. ¿Qué rutas debería exportar AS30 a AS20, y qué rutas no debería exportarle?
12. Teniendo en cuenta las rutas que deben exportarse y las que no, vuelve a configurar BGP en `as20-r1` y `as30-r1` para que se anuncien y se exporten sólo las rutas que a cada AS le interesa.
13. Vuelve a levantar la interfaz `eth0` de `as30-r1` con `ifconfig eth0 up`. Reinicia `quagga` en los 3 *routers* BGP: `as10-r1`, `as20-r1` y `as30-r1`.

14. Comprueba ahora las tablas de encaminamiento en `as20-r1` y `as30-r1`, tanto con `route` como con la interfaz VTY.
15. ¿Qué ruta crees que seguirán ahora los paquetes intercambiados entre `as20-pc3` y `as30-pc2`? Compruébalo.
16. ¿Qué ruta crees que seguirán ahora los paquetes enviados desde `as30-pc3` con destino `as10-pc1`? Compruébalo utilizando `traceroute`. Utilizando la interfaz VTY en `as30-r1` comprueba qué rutas tiene disponibles hacia la red `11.1.0.0/16`.
17. Apaga la interfaz `eth0` de `as30-r1` con `ifconfig eth0 down`. ¿Qué habrá pasado ahora con la ruta que seguirán los paquetes enviados desde `as30-pc3` con destino `as10-pc1`? Compruébalo utilizando `traceroute`. Utilizando la interfaz VTY en `as30-r1` comprueba qué rutas hay ahora para la red `11.1.0.0/16`.

Políticas de exportación y orden de preferencia en la selección de rutas

En el fichero [lab-BGP2.tgz](#) se encuentran los ficheros de configuración para crear un escenario de red como el que se muestra en la siguiente figura.



En esta figura hay 6 AS (Sistemas Autónomos): AS10, AS20, AS30, AS40, AS50 y AS60. Se ha configurado OSPF dentro de AS20, OSPF dentro de AS30 y BGP en todos ellos para intercambiar la información de encaminamiento. Se desea que:

- AS30 y AS10 tengan una relación de tránsito, donde AS30 sea el proveedor y AS10 el cliente.
- AS30 y AS40 tengan una relación de tránsito, donde AS30 sea el proveedor y AS40 el cliente.
- AS40 y AS50 tengan una relación de tránsito, donde AS40 sea el proveedor y AS50 el cliente.
- AS40 y AS60 tengan una relación de tránsito, donde AS40 sea el proveedor y AS60 el cliente.
- AS10 y AS20 tengan una relación de tránsito, donde AS10 sea el proveedor y AS20 el cliente.
- AS20 y AS60 tengan una relación de tránsito, donde AS20 sea el proveedor y AS60 el cliente.
- AS10 y AS40 tengan una relación entre iguales.
- AS20 y AS50 tengan una relación entre iguales.

Arranca todas las máquinas de una en una. Por defecto, al arrancar las máquinas se arranca `quagga`. En el escenario se ha configurado OSPF y BGP. Sin embargo, no se han configurado las políticas de exportación ni el atributo LOCAL PREF de BGP.

1. Piensa en qué *routers* debería existir una lista de exportación de rutas e indica qué rutas deberían estar en dicha lista y a qué *router/s* se le exportaría. Interrumpe `quagga` en los *routers* en los que necesites cambiar la configuración y realiza dicha configuración. Inicia nuevamente `quagga` en dichos *routers*.
2. Comprueba en tu nueva configuración las siguientes reglas consultando cada una de las tablas BGP de los routers de la figura:
 - Un AS no anuncia a su proveedor las subredes aprendidas de otro AS proveedor o de un AS con relación entre iguales.
 - Un AS no anuncia a un AS con relación entre iguales las subredes aprendidas de un AS proveedor o de otro AS con relación entre iguales.
3. Fíjate en la tabla BGP de `as50-r1`. ¿Cuántas rutas hay en la tabla BGP para alcanzar AS60?

4. Interrumpe `quagga` en `as20-r1` . Fíjate en la tabla BGP de `as50-r1` . ¿Cuántas rutas hay en la tabla BGP para alcanzar AS60?
5. ¿Cuál es la ruta que aparece en la tabla de encaminamiento de `as50-r1` para alcanzar AS60?
6. Inicia `quagga` en `as20-r1` . Espera 2 minutos aproximadamente para que `as50-r1` y `as20-r1` hayan intercambiado la información de encaminamiento BGP. ¿Cuántas rutas hay en la tabla BGP para alcanzar AS60?
7. ¿Cuál es la ruta que aparece en la tabla de encaminamiento de `as50-r1` para alcanzar AS60?
8. ¿Es consistente esta ruta con las relaciones entre ASs definidas previamente?
9. Piensa en los atributos LOCAL PREF que configurarías en `as50-r1` y realiza dicha configuración en el escenario. Reinicia `quagga` en `as50-r1` .
10. Interrumpe `quagga` nuevamente en `as20-r1` . Fíjate en la tabla BGP de `as50-r1` . ¿Cuántas rutas hay en la tabla BGP para alcanzar AS60?
11. Inicia `quagga` en `as20-r1` . Espera 2 minutos aproximadamente para que `as50-r1` y `as20-r1` hayan intercambiado la información de encaminamiento BGP. ¿Cuántas rutas hay en la tabla BGP para alcanzar AS60?
12. ¿Cuál es la ruta que aparece en la tabla de encaminamiento de `as50-r1` para alcanzar AS60? Ahora debería ser consistente con las relaciones entre ASs definidas previamente.

Ruta por defecto

1. Cambia la configuración de `as40-r1` de forma que AS40 anuncie a AS50 una ruta por defecto. No elimines aún los anuncios de las subredes individuales.
2. Reinicia `quagga` en `as40-r1` y observa la tabla de encaminamiento y la tabla BGP de `as50-r1` . Comprueba que en ambas tablas hay una ruta por defecto, pero siguen estando las rutas individuales. Las rutas a las subredes individuales, al ser más específicas serán las utilizadas, sin llegar a usarse nunca la ruta por defecto.
3. Cambia la configuración en `as40-r1` para evitar que siga anunciando a AS50 las subredes individuales.
4. Reinicia `quagga` en `as40-r1` y comprueba que la tabla de encaminamiento y la tabla BGP de `as50-r1` ahora sólo tienen la ruta por defecto.

Control de tráfico y DiffServ en Linux

- [Introducción](#)
- [Elementos del control de tráfico](#)
 - [Interfaces de entrada/salida de un router](#)
 - [Qdisc/Class/Filter](#)
- [Configuración de una disciplina de cola a la entrada/salida de un router](#)
 - [Creación de disciplinas de cola a la entrada y a la salida de un router](#)
 - [Descriptor \(*handle*\) de una disciplina de cola](#)
 - [Borrado de disciplinas de cola](#)
- [Control de admisión para el tráfico de entrada \(*policing*\)](#)
- [Control del tráfico de salida](#)
 - [PFIFO para el tráfico de salida](#)
 - [PRIO para el tráfico de salida](#)
 - [Token Bucket Filter \(TBF\) para el tráfico de salida](#)
 - [TBF y PRIO](#)
 - [Hierarchical Token Bucket \(HTB\) para el tráfico de salida](#)
- [DiffServ](#)
 - [Marcado del tráfico DiffServ en la cabecera IPv4](#)
 - [Disciplina de cola DSMARK](#)
 - [Clases de tráfico](#)
 - [Uso del filtro `tcindex` en el Router Frontera](#)
 - [Uso del filtro `tcindex` en el Router de Núcleo](#)
- [Iperf: Generador de tráfico](#)
- [Wireshark](#)

Introducción

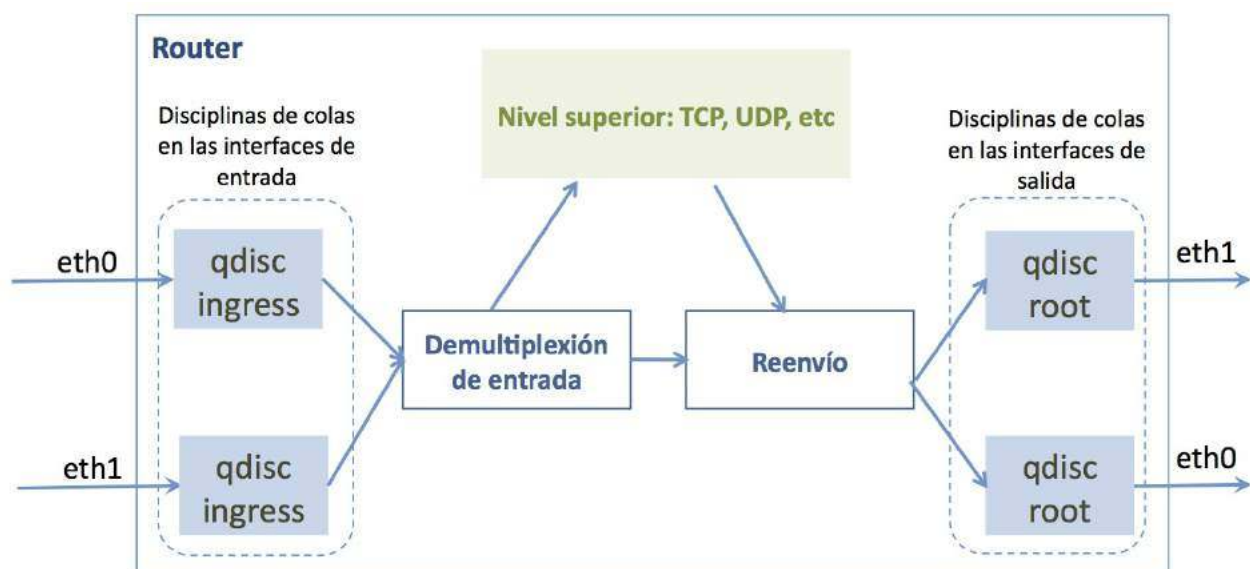
- `tc` (*Traffic Control*) es una herramienta que permite realizar el control de tráfico en Linux.
- `tc` utiliza disciplinas de colas, denominadas `qdisc` (*queueing discipline*).
- Cada vez que el kernel tiene que enviar un paquete a una interfaz, se encola en la `qdisc` configurada en dicha interfaz. El kernel trata de sacar el máximo número de paquetes de dicha `qdisc` para entregárselos al driver de la tarjeta de red.

- El tipo de `qdisc` más simple es FIFO (First In First Out). Si el driver de la tarjeta de red está ocupado, el paquete se quedará guardado en la cola.

Elementos del control de tráfico

Interfaces de entrada/salida de un router

- El control de tráfico se puede aplicar en la interfaz de entrada de un router o en la interfaz de salida del router.
- Cada interfaz de red de un router puede actuar como entrada de paquetes, para los paquetes que se reciben en dicha interfaz, y como interfaz de salida, para los paquetes que se envían a través de dicha interfaz.



Qdisc/Class/Filter

- Qdisc (Disciplina de cola): determina qué paquetes se reenviarán antes que otros. Las hay de 2 tipos:

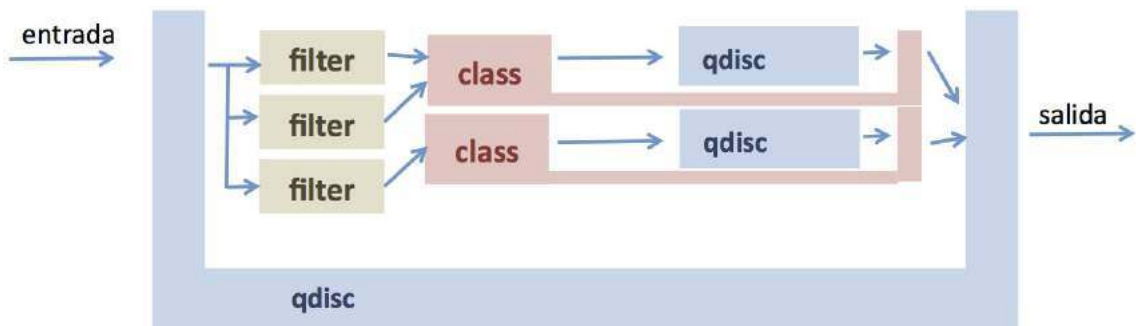
- Disciplina de colas **sin clases de tráfico**: ejemplo FIFO.



- Disciplina de colas **con clases de tráfico**: ejemplo PRIO

- Class (Clase de tráfico): especifica las diferentes categorías de tráfico. Una clase de tráfico está asociada a una disciplina de cola (`qdisc`).

- Filter (Filtro de tráfico): identifica el tráfico que se corresponde con cada clase (class).



Configuración de una disciplina de cola a la entrada/salida de un router

Creación de disciplinas de cola a la entrada y a la salida de un router

- En una determinada interfaz se pueden definir:
 - **Disciplinas de cola de entrada:** para los paquetes que entran a través de dicha interfaz.

```
tc qdisc add dev <interfaz> ... ingress
```

- **Disciplinas de cola de salida:** para los paquetes que salen a través de dicha interfaz.

```
tc qdisc add dev <interfaz> root ...
```

- Por defecto, si no se modifican las disciplinas de cola a la entrada y a la salida de un router, el router aplicará una variante de FIFO (`pfifo_fast` , FIFO mejorado).

Descriptor (*handle*) de una disciplina de cola

- Cuando se define una disciplina de cola se le asocia un descriptor (*handle*) para poder referenciarla.
- El *handle* está formado por 2 números separados por el caracter “:”, es decir, X:Y
 - X : es el número mayor.

- Y : es el número menor.
- Típicamente el *handle* de la disciplina de la cola de entrada (*ingress*) es `ffff:` lo que equivale a `ffff:0` .

```
tc qdisc add dev <interfaz> ingress handle ffff:
```

- Típicamente el *handle* de la disciplina de la cola de salida (*root*) es `1:` lo que equivale a `1:0` .

```
tc qdisc add dev <interfaz> root handle 1: ...
```

Borrado de disciplinas de cola

- El borrado de disciplinas de cola en la entrada y en la salida de un router se hace de la siguiente forma:
- Borrado de la disciplina de la cola de entrada:

```
tc qdisc del dev <interfaz> ingress
```

- Borrado de la disciplina de la cola de salida:

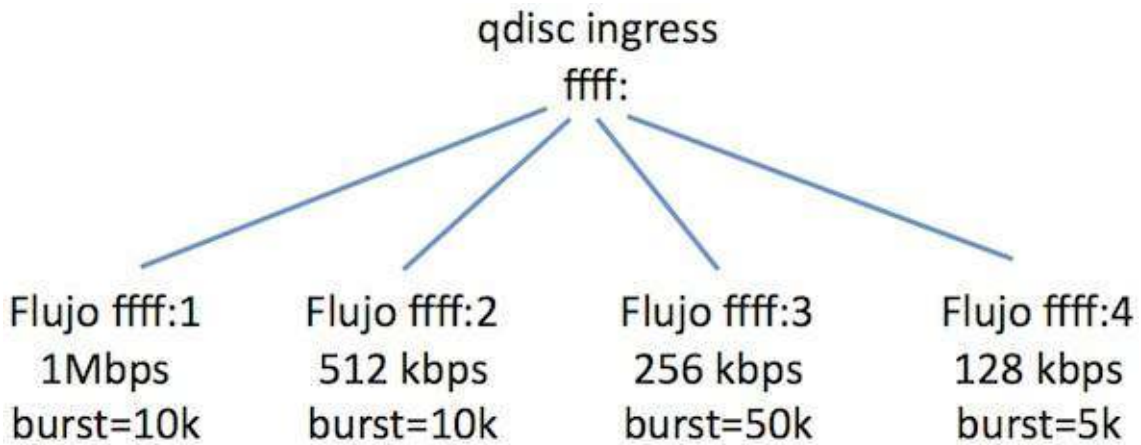
```
tc qdisc del dev <interfaz> root
```

Control de admisión para el tráfico de entrada (*policing*)

- El control de admisión para el tráfico de entrada se utiliza para garantizar que el router sólo procesa para un determinado flujo de paquetes de entrada un máximo de ancho de banda.
- Para garantizar que se cumple el control de admisión el router debe definir una disciplina de colas en la interfaz de red asociada a la entrada de paquetes. Ejemplo, si los paquetes se reciben en la interfaz `eth0` :

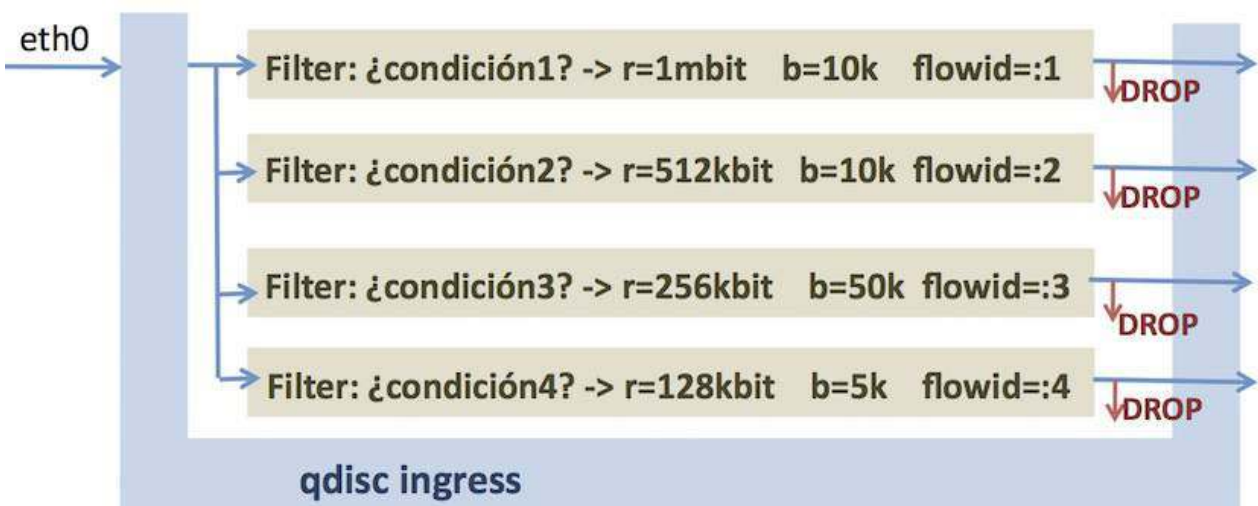
```
tc qdisc add dev eth0 handle ffff: ingress
```

- Se definen filtros para cada uno de los flujos de entrada que van a pertenecer a esa cola de entrada en la interfaz `eth0` (`handle ffff:`) indicando el ancho de banda máximo y el tamaño de la cubeta. El control de admisión se basa en el modelo TBF. Si el tráfico de entrada supera esa especificación se descarta o se le pasa al siguiente filtro.



Ejemplo:

- Se define `qdisc` para la entrada.
- Se definen filtros para clasificar el tráfico: los paquetes que cumplen una condición se les aplica un TBF y quedan clasificados dentro de un determinado flujo (`:`):
 - **Condición:** utilizaremos el que comprueba campos de la cabecera IP de los paquetes.
 - **Perfil de tráfico:** se configura con `.` El tráfico que supere la especificación puede reclasificarse (`continue`) o descartarse (`drop`).
- Los filtros se aplican siguiendo el orden dado por el parámetro prioridad: primero los filtros cuyo valor de prioridad sea menor (números de prioridad bajos equivale a mayor prioridad).



```
tc qdisc add dev eth0 handle ffff: ingress

tc filter add dev eth0 parent ffff: \
  protocol ip prio 4 u32 \
  match ip src 11.0.0.100/32 \
  police rate 1mbit burst 10k continue flowid :1

tc filter add dev eth0 parent ffff: \
  protocol ip prio 5 u32 \
  match ip src 11.0.0.100/32 \
  police rate 512kbit burst 10k continue flowid :2

tc filter add dev eth0 parent ffff: \
  protocol ip prio 6 blue u32 \
  match ip src 11.0.0.100/32 \
  police rate 256kbit burst 50k drop flowid :3

tc filter add dev eth0 parent ffff: \
  protocol ip prio 6 u32 \
  match ip src 0.0.0.0/0 \
  police rate 128kbit burst 5k drop flowid :4
```

Control del tráfico de salida

PFIFO para el tráfico de salida

- FIFO con tamaño de cola dado por el valor `limit` en número de paquetes:

```
tc qdisc add dev eth0 root pfifo limit 5
```

- Es una disciplina de colas muy sencilla que requiere poco tiempo de computación.
- Si hay congestión, FIFO perjudica a TCP debido a que una pérdida de un paquete en TCP activa los mecanismos de control de congestión que provocan una disminución de la tasa de envío para la conexión TCP. Una pérdida de un paquete UDP no tiene ningún impacto en la tasa de envío para esta comunicación.
- Una ráfaga de uno de los flujos que atravesase una cola FIFO puede llenar la cola y perjudicar al resto de los flujos.

PRIO para el tráfico de salida

- PRIO es una disciplina de cola de prioridades que realiza algunas acciones automáticamente. El siguiente comando crea la disciplina de cola 1:0 y automáticamente crea las clases 1:1, 1:2 y 1:3 de tipo PFIFO.

```
tc qdisc add dev eth0 root handle 1: prio
```

![image](./tc/figs/qos-14.png)

- Si se desea crear un número diferente de prioridades se puede usar el parámetro:

```
bands <núm_prio> .
```

- Es necesario especificar los filtros para clasificar el tráfico dentro de las clases 1:1, 1:2 y 1:3 del ejemplo. Esta clasificación puede hacerse a través del filtro u32 utilizando la dirección IP origen:

```
tc filter add dev eth0 parent 1:0 prio 1 protocol ip u32 \  
    match ip src 11.0.0.1/32 flowid 1:1  
tc filter add dev eth0 parent 1:0 prio 2 protocol ip u32 \  
    match ip src 11.0.0.2/32 flowid 1:2  
tc filter add dev eth0 parent 1:0 prio 3 protocol ip u32 \  
    match ip src 11.0.0.3/32 flowid 1:3
```

- La disciplina de colas de prioridad puede provocar retrasos y pérdidas de paquetes en la clase menos prioritaria 1:3 si hay mucho tráfico en 1:1 y 1:2.
- Para solventar que TCP se vea perjudicado con PFIFO, se podría clasificar tráfico TCP en 1:1 y el UDP en 1:2. Sin embargo, si hubiera muchos datos para TCP, los mecanismos de control de congestión de TCP provocarían que la tasa de envío fuera aumentando cada vez más mientras no se perdieran paquetes, viéndose en este caso UDP perjudicado ya que mientras hubiera tráfico en 1:1 no se cursaría el resto de tráfico.

Token Bucket Filter (TBF) para el tráfico de salida

- TBF limita la velocidad del tráfico de salida de una determinada interfaz de un router a través de los parámetros:
 - ancho de banda: velocidad de generación tokens
 - tamaño de la cubeta: para almacenar tokens que permitirán enviar ráfagas que superen el ancho de banda.
 - latencia: tiempo máximo de espera de un paquete por un token.

- Ejemplo: Aplicar TBF en la interfaz de salida eth1 de un router. Ancho de banda 7Mbps , tamaño de cubeta 10k (en bytes), latencia 50 ms:

```
tc qdisc add dev eth1 handle 1: root tbf rate 7Mbit \  
burst 10k latency 50ms
```

TBF y PRIO

- TBF limita la velocidad del tráfico de salida de una determinada interfaz para todos los flujos que lo atraviesan.
- Se puede definir una disciplina de cola PRIO hija de TBF para que el tráfico además de estar limitado por TBF los paquetes se cursen atendiendo a las prioridades definidas en PRIO:

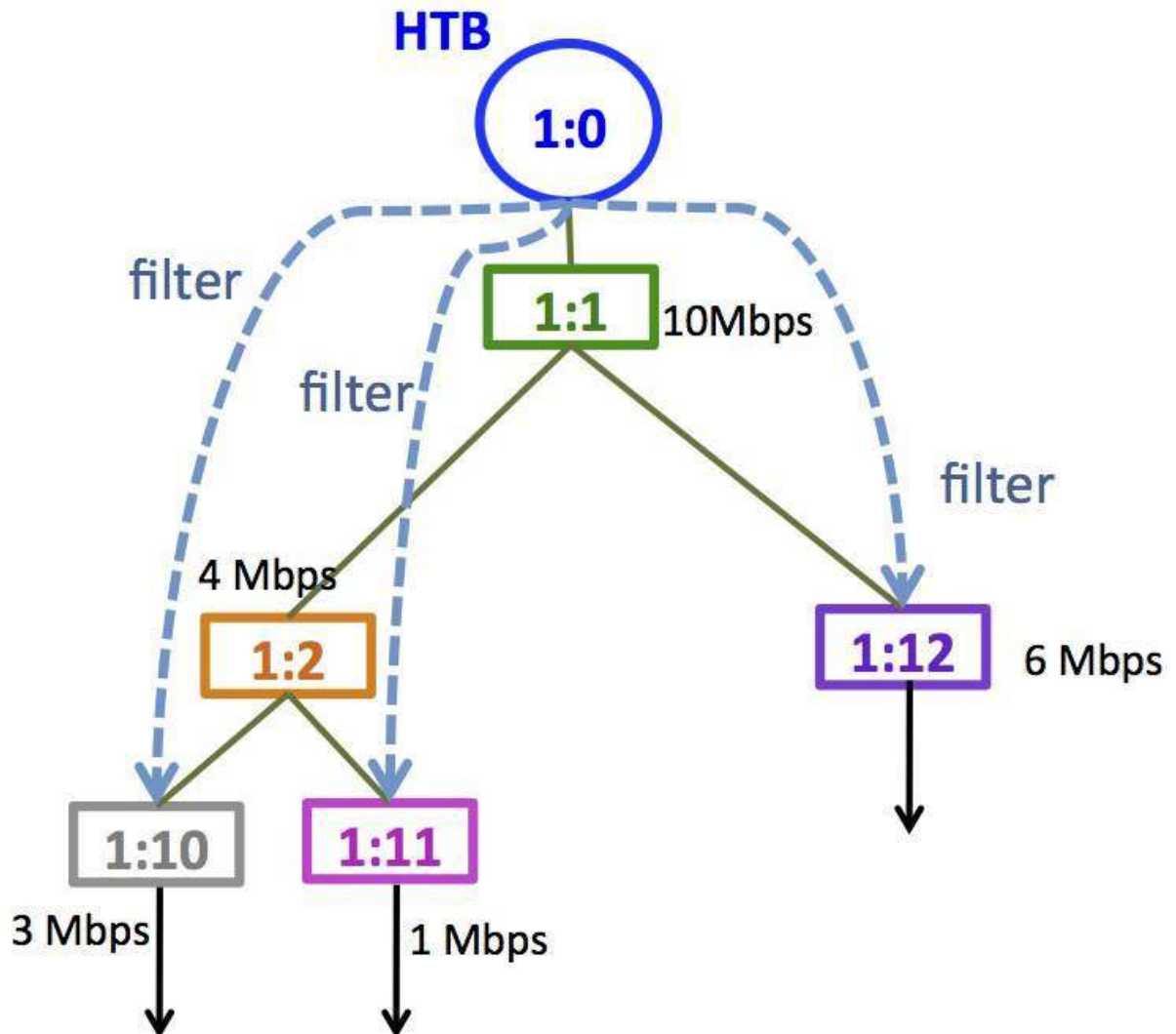
```
tc qdisc add dev eth1 handle 1: root tbf rate 7Mbit \  
burst 10k latency 50 ms  
  
tc qdisc add dev eth1 parent 1:0 handle 10:0 prio  
  
tc filter add dev eth1 parent 10:0 prio 1 protocol ip u32 \  
match ip src 101.0.0.10/32 flowid 10:1  
tc filter add dev eth1 parent 10:0 prio 2 protocol ip u32 \  
match ip src 102.0.0.20/32 flowid 10:2  
tc filter add dev eth1 parent 10:0 prio 3 protocol ip u32 \  
match ip src 103.0.0.30/32 flowid 10:3
```

Hierarchical Token Bucket (HTB) para el tráfico de salida

- TBF se utiliza cuando se desea que todo el tráfico que sale por una determinada interfaz tenga unas características determinadas en cuanto a ancho de banda, tamaño de cubeta y latencia.
- HTB es necesario cuando se quiere clasificar el tráfico que sale por una determinada interfaz en varias clases, cada uno de ellas con unas características diferentes de ancho de banda.
- Si alguna clase no utiliza todo el ancho de banda que se le ha definido, dependiendo de la configuración, se podría utilizar dicho ancho de banda para algún otra clase que lo necesite.

Ejemplo:

- Definición de HTB con ancho de banda máximo de salida de 10Mbps de la interfaz eth0. Se quiere repartir este ancho de banda:
 - usuarioA (4Mbps): lo usará para tráfico HTTP (3Mbps) y tráfico de correo (1Mbps)
 - el usuarioB (6Mbps)



```

...
tc qdisc add dev eth0 root handle 1:0 htb

tc class add dev eth0 parent 1:0 classid 1:1 htb rate 10Mbit

tc class add dev eth0 parent 1:1 classid 1:2 htb rate 4Mbit ceil 10Mbit
tc class add dev eth0 parent 1:2 classid 1:10 htb rate 3Mbit ceil 10Mbit
tc class add dev eth0 parent 1:2 classid 1:11 htb rate 1Mbit ceil 10Mbit
tc class add dev eth0 parent 1:1 classid 1:12 htb rate 6Mbit ceil 10Mbit

tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
  match ip src 1.1.1.1 \
  match ip dport 80 0xffff \
  flowid 1:10

tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
  match ip src 1.1.1.1 \
  match ip dport 25 0xffff \
  flowid 1:11

tc filter add dev eth0 parent 1:0 protocol ip prio 1 u32 \
  match ip src 2.2.2.2 \
  flowid 1:12
...

```

Unidades para tc

- Ancho de banda

Abreviatura | Unidades -----|----- kbps | Kilobytes per second mbps |
 Megabytes per second kbit | Kilobits per second mbit | Megabits per second bps | Bytes
 per second

- Cantidad de datos

Abreviatura | Unidades -----|----- kb o k | Kilobytes mb o m | Megabytes
 kbit | Kilobits mbit | Megabits b | Bytes

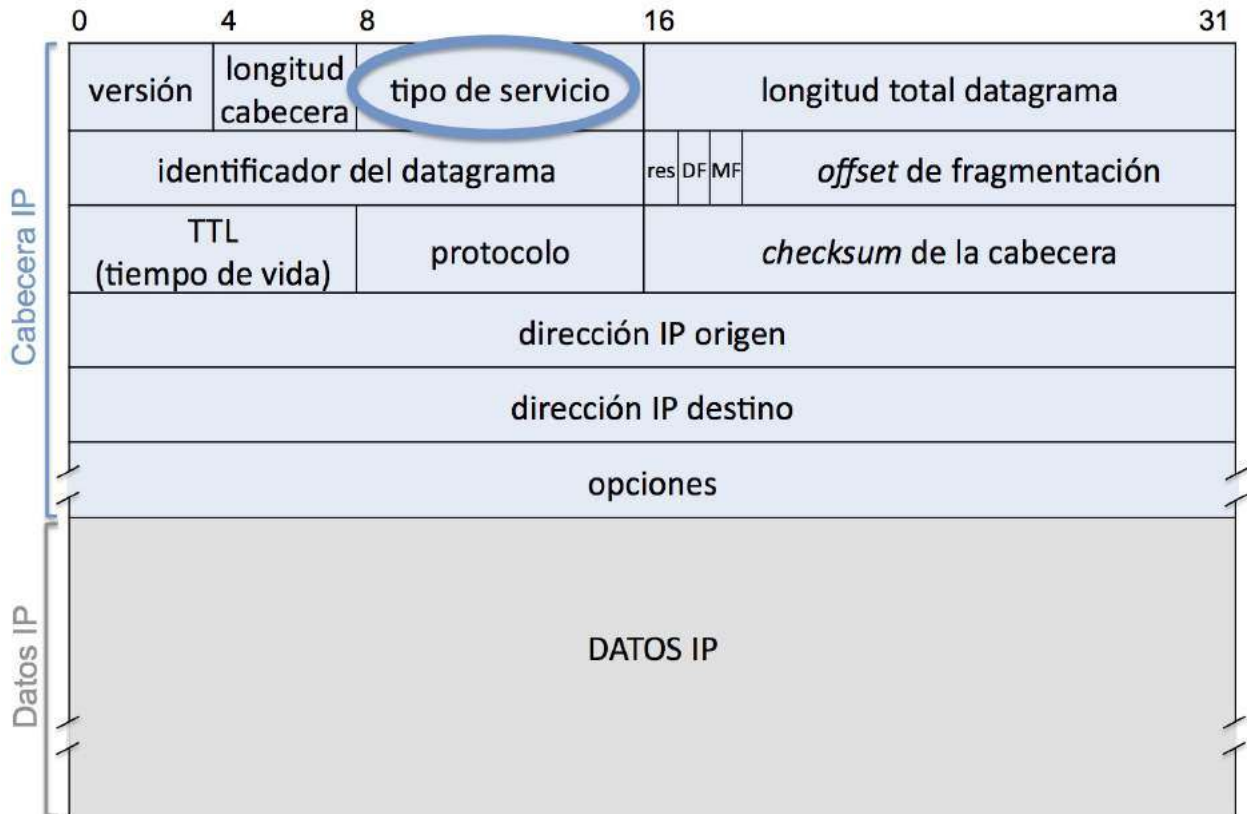
- Tiempo

Abreviatura | Unidades -----|----- s, sec o secs | Segundos ms,
 msec o msecs | Milisegundos us, usec, usecs | Microsegundos

DiffServ

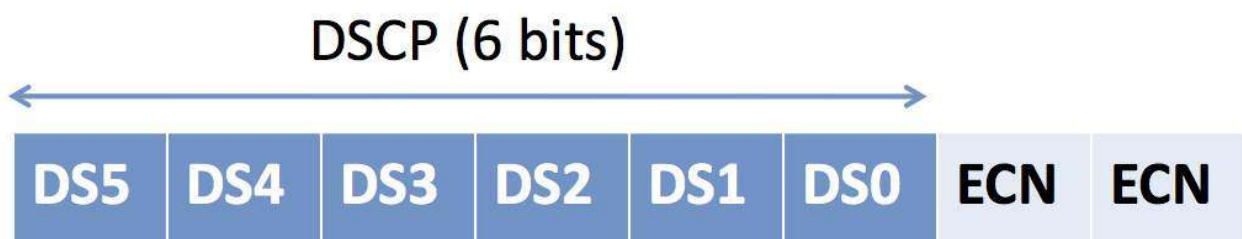
Marcado del tráfico DiffServ en la cabecera IPv4

- Los paquetes se marcan en el campo de 8 bits Type of Service (ToS) de IPv4



Marcado del tráfico DiffServ: campo DSCP

- Se usan 6 bits para identificar *Differentiated Service Code Point (DSCP)* que determinan el comportamiento por salto (PHB, Per-Hop Behavior) que recibirá el paquete en los *routers* de la red DiffServ.
- Quedan los 2 bits menos significativos del campo ToS que no se usan para DiffServ, sino para la notificación de congestión (Explicit Congestion Notification, ECN). ECN es utilizado conjuntamente por los extremos de una conexión TCP y los routers intermedios que usan la disciplina de cola RED, Random Early Detection.




Comportamiento por salto (PHB)

- Cada clase de tráfico (DSCP) está asociado a un comportamiento por salto (*PHB, Per Hop Behavior*). El PHB determina el tipo de tratamiento que se le va a dar al paquete en el reenvío:
 - Reserva de recursos: buffer y ancho de banda.
 - Características del tráfico: retardo y pérdidas.
- DiffServ no especifica las características concretas del tráfico en cada una de las clases, sólo proporciona la clasificación con diferentes códigos DSCP.
- Existen los siguientes grupos de PHB asociados al campo DSCP:
 - **EF** (Expedited Forwarding): DSCP=
 - Bajas pérdidas, baja latencia, bajo jitter, similar a una línea de datos alquilada.
 - **VA** (Voice Admit): DSCP=
 - Similar a EF que añade un mecanismo de control de admisión de llamadas.
 - **AF** (Assured Forwarding): 4 clases (AF1, AF2, AF3, AF4)
 - Se proporciona cierta garantía de entrega siempre y cuando se cumpla el acuerdo entre cliente y proveedor sobre el tráfico enviado.
 - Define 4 clases. Prioridad AF4 $\$>\$$ Prioridad AF3 $\$>\$$ Prioridad AF2 $\$>\$$ Prioridad AF1.
 - **DF** (Default Group): DSCP=
 - IP Best Effort (compatible con tráfico que no es DiffServ)
 - **CS** (Class Selector): usa los 3 primeros bits DSCP=000 para definir prioridades (compatible con el antiguo ToS).
 - Menor prioridad = 000 a mayor prioridad = 000

Marcas en los paquetes: Valores DS y DSCP

- El valor DS son 8 bits que viajan en el campo Tipo de Servicio de la cabecera IP. Estos 8 bits son en realidad DSCP (6 bits) + ECN (2 bits).

		DSCP (6bits)	DS (8 bits)	Prioridad de descarte de tráfico	
Clase menos prioritaria  Clase más prioritaria	Clase AF1	AF11	001 010	0010 1000 – 0x28	Baja
		AF12	001 100	0011 0000 – 0x30	Media
		AF13	001 110	0011 1000 – 0x38	Alta
	Clase AF2	AF21	010 010	0100 1000 – 0x48	Baja
		AF22	010 100	0101 0000 – 0x50	Media
		AF23	010 110	0101 1000 – 0x58	Alta
	Clase AF3	AF31	011 010	0110 1000 – 0x68	Baja
		AF32	011 100	0111 0000 – 0x70	Media
		AF33	011 110	0111 1000 – 0x78	Alta
	Clase AF4	AF41	100 010	1000 1000 – 0x88	Baja
		AF42	100 100	1001 0000 – 0x90	Media
		AF43	100 110	1001 1000 – 0x98	Alta
	Clase EF		101 110	1011 1000 – 0xb8	

Disciplina de cola DSMARK

- DSMARK es una qdisc que se utiliza para la clasificación de paquetes según la arquitectura de DiffServ y el marcado del campo DS.
- Probaremos la función de marcado del campo DS:
 - **Routers frontera de DiffServ:** es necesario clasificar los paquetes atendiendo a valores de su cabecera IP y marcar el campo DS.
- Probaremos la función de clasificación según el campo DS:
 - **Routers del núcleo de DiffServ:** Los paquetes se clasifican atendiendo al contenido de DS que traen los paquetes IP para posteriormente realizar su tratamiento según diferentes disciplinas de cola.

Clases de tráfico

- Una qdisc DSMARK crea automáticamente un conjunto de clases para asignar en cada una de ellas un valor diferente en el campo DSCP.
- Al crear la qdisc DSMARK es necesario especificar la cantidad máxima de valores DSCP que se van a usar para clasificar los paquetes. Este valor debe ser una potencia de 2.

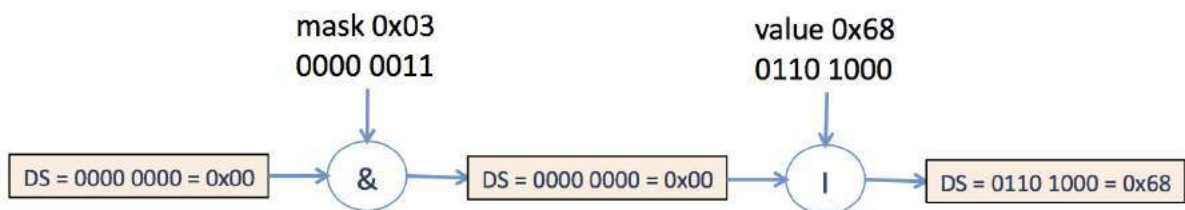
- Por ejemplo, la siguiente qdisc con número de clases igual a $2^3=8$ podrá definir como máximo 7 valores DSCP diferentes, uno para cada una de sus clases:

```
tc qdisc add dev eth1 handle 1:0 root dsmark indices 8
```

- Una vez creada una qdisc DSMARK se habrán creado automáticamente sus clases asociadas. Cada clase tiene un descriptor X:Y, donde:
 - X es el valor del número mayor de la clase qdisc a la que pertenece
 - Y es el número menor que distingue a cada una de las clases
- Por ejemplo, para definir como máximo 7 clases de tráfico dentro de una qdisc se usarán los siguientes descriptores: 1:1, 1:2, 1:3, 1:4, 1:5, 1:6, 1:7. El número mayor coincide en todas ellas, ya que todas pertenecen a la qdisc .

Uso del filtro `tcindex` en el Router Frontera

- Para configurar un valor DS en un paquete IP se utiliza una clase DSMARK en la que se van a realizar las siguientes operaciones:
 1. operación `&` de bits con una máscara para mantener aquellos bits del campo DS que se deseen (0x03, mantiene los 2 bits menos significativos donde viaja ECN)
 2. operación `|` de bits con el nuevo valor DS que se desee asignar.



- Por ejemplo: si la clase 1:3 marca con DS=0x68 (AF31) conservando los bits ECN:

```
tc class change dev eth1 classid 1:3 dsmark mask 0x3 value 0x68
```

- Al recibirse un paquete IP en un router, se almacena en un buffer.
- Cuando un paquete atraviesa la disciplina de cola de entrada `qdisc ingress`, el buffer que almacena el paquete IP dentro del kernel del router Linux tiene un campo `tc_index` en el que se almacena el identificador de flujo en el que fue clasificado dicho paquete cuando ingresó en la cola `qdisc ingress`:
 - si un paquete queda clasificado en `flowid :2`, el campo `tc_index` almacenaría el

valor 2.

- Esta información es la que utiliza posteriormente el filtro tcindex dentro de otras qdisc de dicho router para dar tratamiento diferenciado a los paquetes.

```

...
tc qdisc add dev eth0 handle ffff: ingress

tc filter add dev eth0 parent ffff: \
  protocol ip prio 4 u32 \
  match ip src 11.0.0.100/32 \
  police rate 1mbit burst 10k continue flowid :1

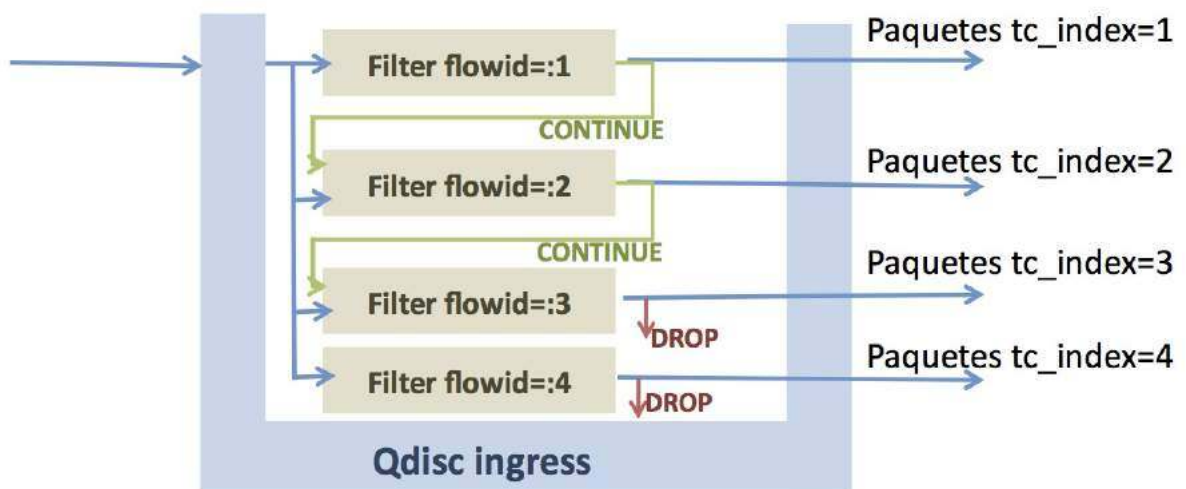
tc filter add dev eth0 parent ffff: \
  protocol ip prio 5 u32 \
  match ip src 11.0.0.100/32 \
  police rate 512kbit burst 10k continue flowid :2

tc filter add dev eth0 parent ffff: \
  protocol ip prio 6 u32 \
  match ip src 11.0.0.100/32 \
  police rate 256kbit burst 10k drop flowid :3

tc filter add dev eth0 parent ffff: \
  protocol ip prio 6 u32 \
  match ip src 0.0.0.0/0 \
  police rate 128kbit burst 10k drop flowid :4
...

```

- Cuando un paquete atraviesa la disciplina de cola de entrada, el buffer que contiene el paquete tiene un campo donde se ha almacenado el descriptor del flujo al que pertenece: `tc_index`.



- Ejemplo: Los paquetes clasificados dentro del `flowid=:1` llevarán en el campo `tc_index=1`

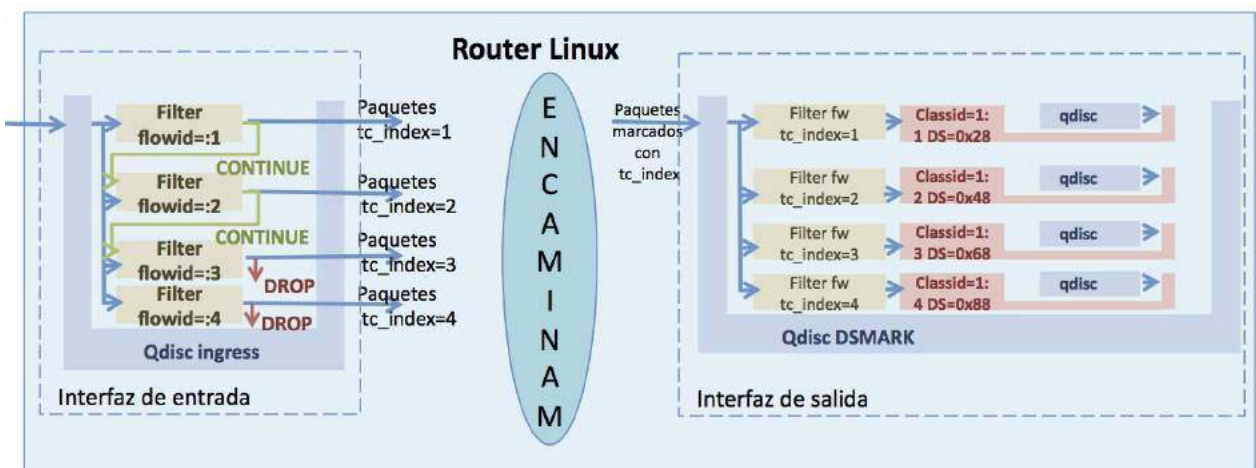
- EL filtro utiliza el valor almacenado en `tc_index` del buffer que contiene un paquete IP para clasificarlo dentro de una clase en la disciplina de cola de salida, en este caso DSMARK.
- Ejemplo:
 - `qdisc DSMARK con 3 diferentes valores de marcado`

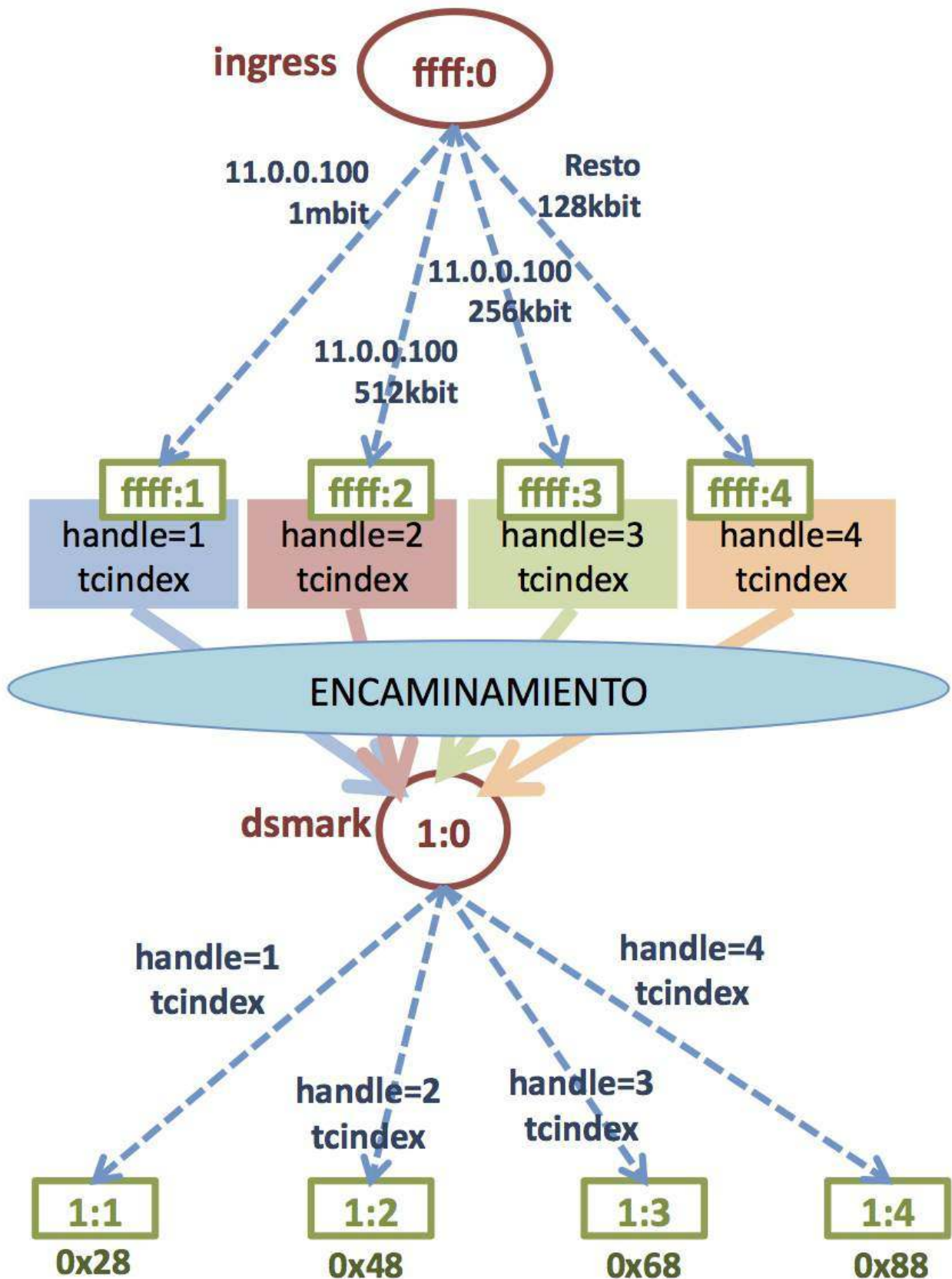
```
tc qdisc add dev eth1 handle 1:0 root dsmark indices 8

tc class change dev eth1 classid 1:1 dsmark mask 0x3 value 0x28
tc class change dev eth1 classid 1:2 dsmark mask 0x3 value 0x48
tc class change dev eth1 classid 1:3 dsmark mask 0x3 value 0x68
tc class change dev eth1 classid 1:4 dsmark mask 0x3 value 0x88

tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
    handle 1 tcindex classid 1:1
tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
    handle 2 tcindex classid 1:2
tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
    handle 3 tcindex classid 1:3
tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
    handle 4 tcindex classid 1:4
```

- Los paquetes entran en el router y atraviesan la cola ingresa, donde se rellena el campo `tc_index`.
- Dentro de la cola DSMARK en la interfaz de salida, se comprueba el valor de `tc_index` y se clasifica el tráfico en las clases de DSMARK, que marcarán los paquetes con un determinado valor de DS.





Uso del filtro `tcindex` en el Router de Núcleo

- En el **router del núcleo**, el paquete llega con DS relleno.

- Si el router no tiene configurada una disciplina de cola ingress, inicialmente no se rellena el campo `tc_index` del buffer del kernel del Linux que contiene el paquete.
- La propia qdisc DSMARK puede copiar el campo DS que lleva la cabecera IP del paquete en el campo `tc_index` de la estructura de datos del kernel del Linux que almacena el paquete utilizando el siguiente parámetro:

```
tc qdisc add dev eth1 handle 1:0 root dsmark indices 8 set_tc_index
```

- Como se copia el campo DS, el filtro deberá utilizar sólo los 6 bits más significativos del campo DS, es decir, DSCP para clasificar los paquetes. Es necesario extraer esos 6 bits (operación `&` con y desplazamiento a la derecha 2 bits) del campo DSCP:

```
tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
    tcindex mask 0xfc shift 2
```

- A continuación se puede usar otra disciplina de cola encadenada con DSMARK para realizar la planificación de paquetes que se desee, por ejemplo HTB:

```
tc qdisc add dev eth1 parent 1:0 handle 2:0 htb

tc class add dev eth1 parent 2:0 classid 2:1 htb rate 1Mbit
tc class add dev eth1 parent 2:1 classid 2:10 htb rate 800kbit ceil 1Mbit

# Tráfico AF11=> DS=0x28 =00101000 => DSCP=001010=0x0a
tc filter add dev eth1 parent 2:0 protocol ip prio 1 \
    handle 0x0a tcindex classid 2:10
```

- Ejemplo completo:

```
tc qdisc add dev eth1 handle 1:0 root dsmark indices 8 set_tc_index

tc filter add dev eth1 parent 1:0 protocol ip prio 1 \
    tcindex mask 0xfc shift 2

tc qdisc add dev eth1 parent 1:0 handle 2:0 htb

tc class add dev eth1 parent 2:0 classid 2:1 htb rate 1Mbit
tc class add dev eth1 parent 2:1 classid 2:10 htb rate 800kbit ceil 1Mbit

# Tráfico AF11=> DS=0x28 =00101000 => DSCP=001010=0x0a
tc filter add dev eth1 parent 2:0 protocol ip prio 1 \
    handle 0x0a tcindex classid 2:10
```

Iperf: Generador de tráfico

- Iperf es una aplicación que permite generar tráfico TCP o UDP y medir el ancho de banda que se está utilizando entre dos máquinas finales.
- Iperf funciona en modo cliente/servidor.
- Utilizaremos Iperf para generar tráfico UDP desde el cliente al servidor durante 10 segundos a una determinada velocidad y medir el ancho de banda que en realidad se ha utilizado.
- Arrancaremos Iperf de la siguiente forma:
 - Servidor

```
iperf -u -s
```

- Cliente

```
iperf -u -c <dirIPServidor> -b <anchoDeBanda>
```

Resultado de la ejecución de Iperf

- Resultado de ejecución de Iperf en 12.0.0.30, en modo servidor:

```
iperf -s -u
-----
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 108 KByte (default)
-----
[3] local 12.0.0.30 port 5001 connected with 11.0.0.10 port 32768
[3] 0.0- 9.7 sec 25.2 MBytes 21.8 Mbits/sec 0.289 ms 4940/22916 (22%)
[3] 0.0- 9.7 sec 1 datagrams received out-of-order
```

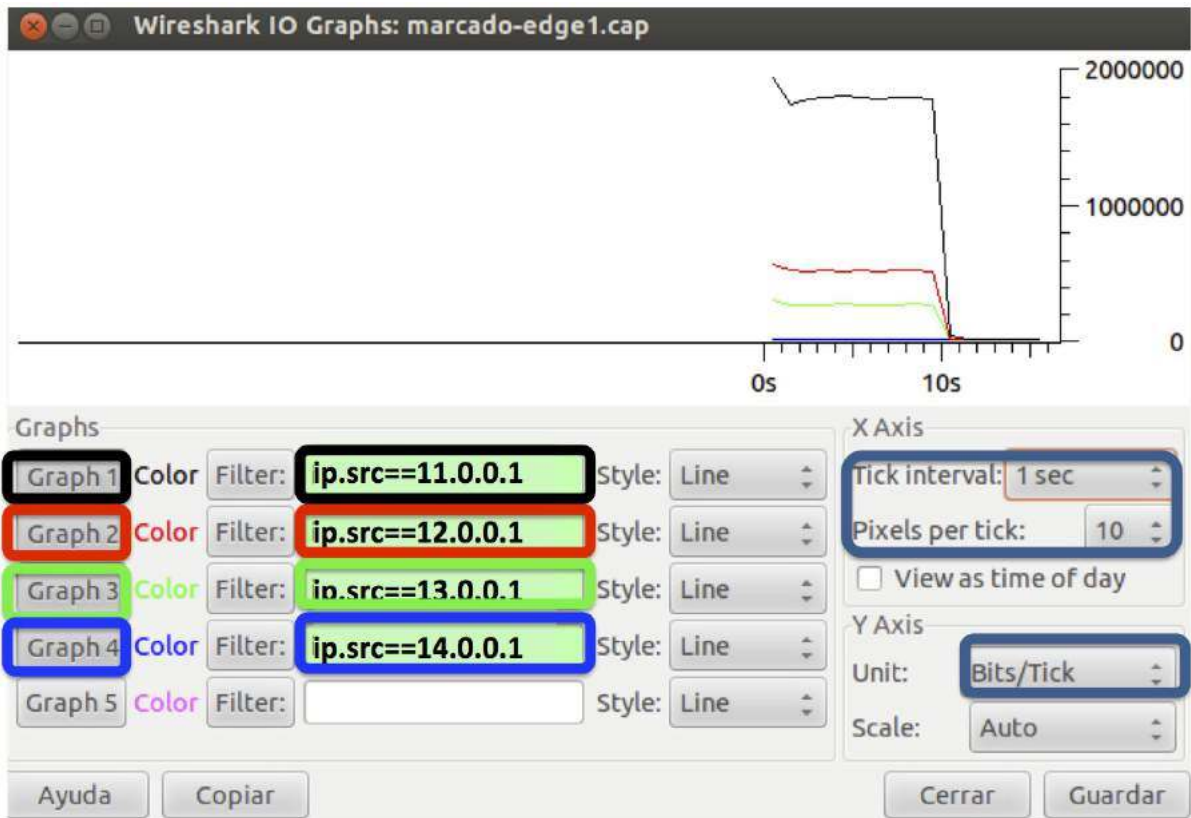
- Resultado de ejecución de Iperf en 11.0.0.10, en modo cliente:

```
iperf -u -c 12.0.0.30 -b 100M
-----
Client connecting to 12.0.0.30, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 108 KByte (default)
-----
[3] local 11.0.0.10 port 32768 connected with 12.0.0.30 port 5001
[3] 0.0- 10.0 sec 32.1 MBytes 26.9 Mbits/sec
[3] Sent 22917 datagrams
[3] Server Report:
[3] 0.0- 9.7 sec 25.2 MBytes 21.8 Mbits/sec 0.289 ms 4940/22916 (22%)
[3] 0.0- 9.7 sec 1 datagrams received out-of-order
```

Wireshark

Gráficas de ancho de banda (I)

- Seleccionar en el menú de Wireshark `Statistics` $\$-\$$ `IO Graphs` .
- Especificar en la casilla `Filter` , el filtro de Wireshark que selecciona los paquetes según los criterios que se deseen. Ejemplo en Filter1: `ip.src==11.0.0.1` . Pulsar sobre el botón `Graph 1` para visualizar la gráfica.
- Para distinguir el tráfico de varias fuentes la condición de filtrado puede completarse con la comprobación de la dirección IP y puerto TCP/UDP. Ejemplo: `ip.src==11.0.0.1 and udp.dstport==5001`
- Si no se escribe nada en la condición de filtrado, se muestra la gráfica de todo el tráfico capturado.
- Es importante seleccionar adecuadamente las unidades que muestra la gráfica. Para las pruebas que vamos a realizar en las prácticas se recomienda:
 - X Axis -> Tick interval: 1 sec
 - X Axis -> Pixels per tic: 10
 - Y Axis -> Unit: Bits/Tic
 - Y Axis -> Scale: Auto



Campo DS en una captura de Wireshark

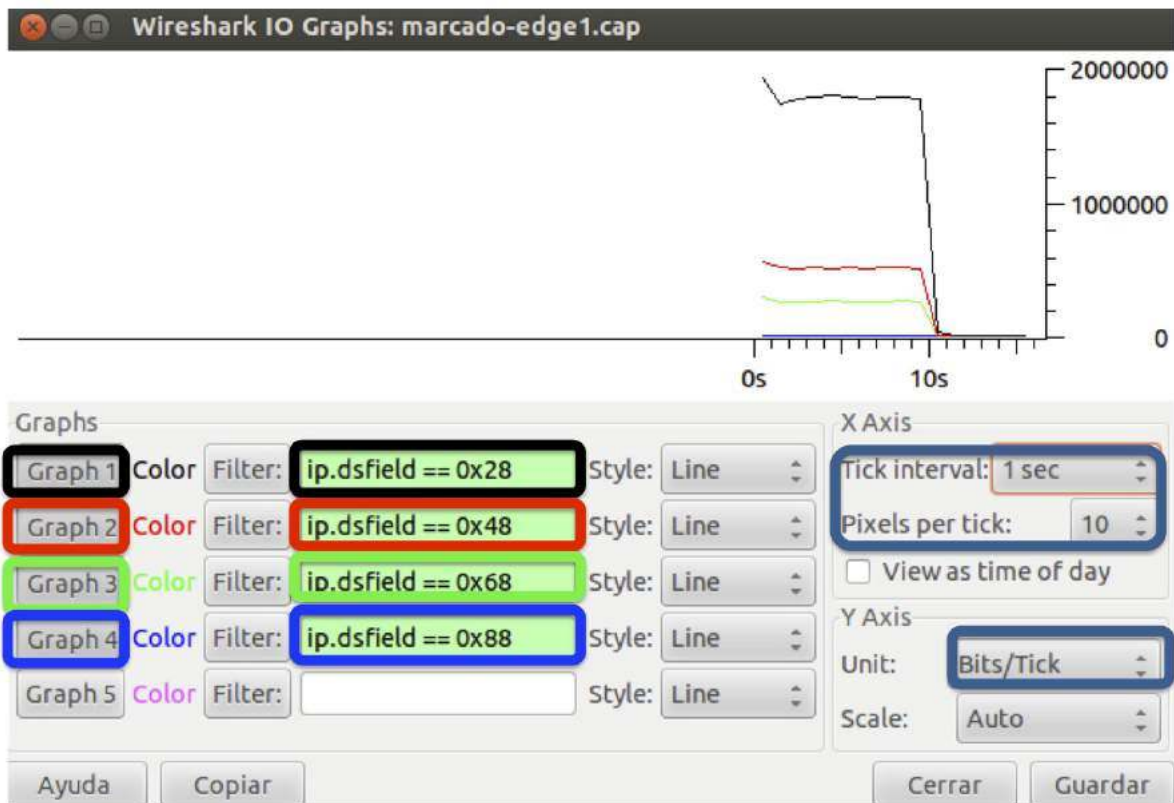
The figure shows a Wireshark packet capture of 'mercado-edge1.cap' [Wireshark 1.6.7]. The packet list shows a packet from 'Navegador web Firefox' (11.0.0.10) to 12.0.0.2. The packet details pane shows the 'Internet Protocol Version 4' section with the 'Differentiated Services Field' highlighted in red. The field value is 0x28, which is decoded as 'DSCP 0x0a: Assured Forwarding 11'. The packet is a UDP packet with source port 33541 and destination port 5001.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.013050	11.0.0.10	12.0.0.2	UDP	1512	Source port: 33541 Destination port: 5001
4	0.017385	11.0.0.10	12.0.0.2	UDP	1512	Source port: 33541 Destination port: 5001
5	0.026378	11.0.0.10	12.0.0.2	UDP	1512	Source port: 33541 Destination port: 5001

▶ Frame 1: 1512 bytes on wire (12096 bits), 1512 bytes captured (12096 bits)
 ▶ Ethernet II, Src: a6:41:79:38:e8:42 (a6:41:79:38:e8:42), Dst: 9a:b8:8b:9b:0c:89 (9a:b8:8b:9b:0c:89)
 ▼ Internet Protocol Version 4, Src: 11.0.0.10 (11.0.0.10), Dst: 12.0.0.2 (12.0.0.2)
 Version: 4
 Header length: 20 bytes
Differentiated Services Field: 0x28 (DSCP 0x0a: Assured Forwarding 11; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
 0010 10.. = Differentiated Services Codepoint: Assured Forwarding 11 (0x0a)
00 = Explicit Congestion Notification: Not-ECT (Not ECN-Capable Transport) (0x00)
 Total Length: 1498
 Identification: 0x266f (9839)
 ▶ Flags: 0x02 (Don't Fragment)
 Fragment offset: 0
 Time to live: 63
 Protocol: UDP (17)
 ▶ Header checksum: 0xf870 [correct]
 Source: 11.0.0.10 (11.0.0.10)
 Destination: 12.0.0.2 (12.0.0.2)
 ▶ User Datagram Protocol, Src Port: 33541 (33541), Dst Port: 5001 (5001)

Gráficas de ancho de banda en diferentes clases DiffServ

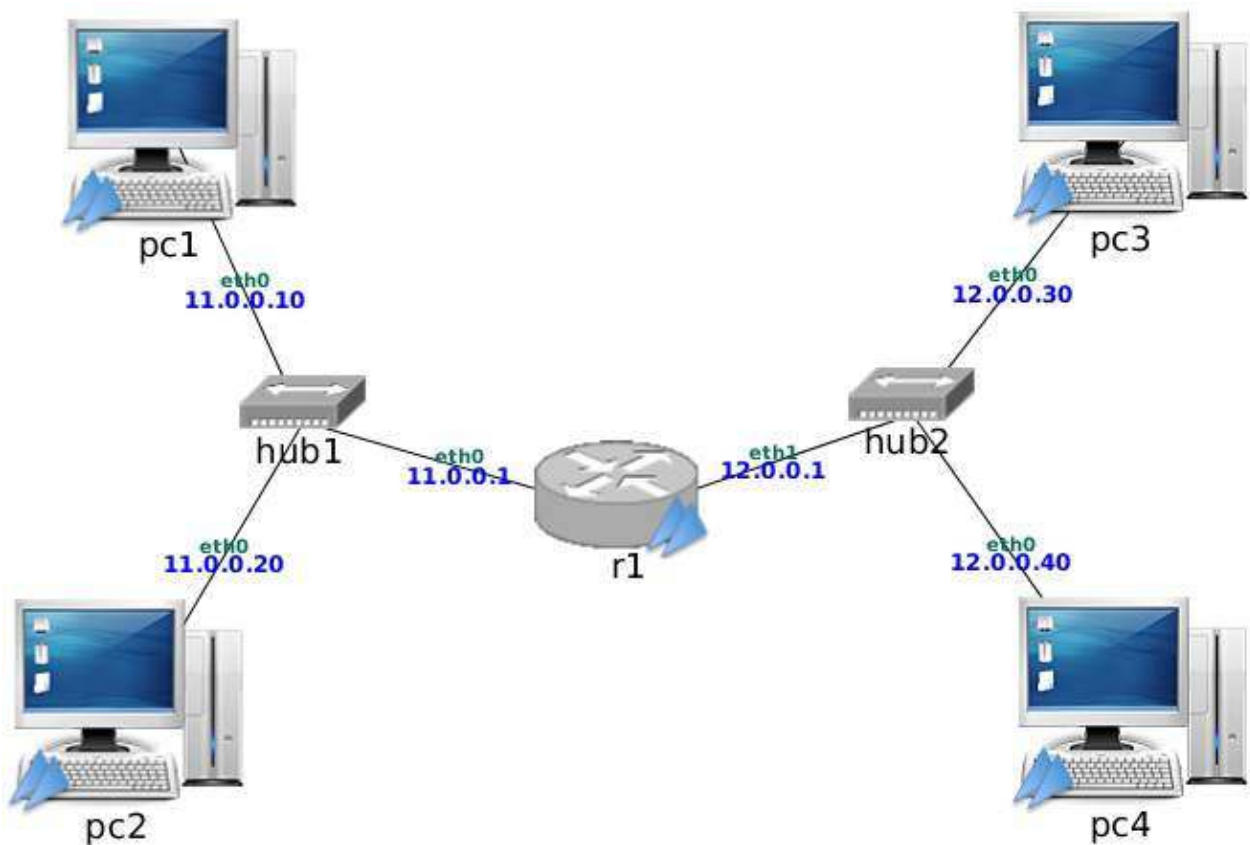
- Seleccionar en el menú de Wireshark `Statistics` \rightarrow `IO Graphs`.
- Especificar en la casilla `Filter`, el filtro de Wireshark que selecciona los paquetes marcados con diferentes DSCP. Ejemplo en Filter1: `ip.dscp==0x28`. Pulsar sobre el botón `Graph 1` para visualizar la gráfica.
- Para distinguir el tráfico de varias fuentes la condición de filtrado puede completarse con la comprobación de la dirección IP. Ejemplo: `ip.dscp==0x28 and ip.src==11.0.0.1`
- Si no se escribe nada en la condición de filtrado, se muestra la gráfica de todo el tráfico capturado.
- Es importante seleccionar adecuadamente las unidades que muestra la gráfica. Para las pruebas que vamos a realizar en las prácticas se recomienda:
 - X Axis \rightarrow Tick interval: 1 sec
 - X Axis \rightarrow Pixels per tic: 10
 - Y Axis \rightarrow Unit: Bits/Tic
 - Y Axis \rightarrow Scale: Auto



Laboratorio de Control de Tráfico

- Sin control de tráfico
 - Un flujo de datos
 - Dos flujos de datos
- Control de admisión para el tráfico de entrada
- Disciplinas de colas para el tráfico de salida
 - Token Bucket Filter (TBF)
 - Prioridad (PRIO)
 - Hierarchical Token Bucket (HTB)
- DiffServ
 - Configuración de función policing y marcado de tráfico en DSCP
 - Tratamiento de tráfico en función del marcado DSCP

Descomprime el fichero que contiene el escenario de NetGUI [lab-tc.tgz](#) para realizar la práctica de control de tráfico en Linux.



Sin control de tráfico

El *router* `r1` no tiene activado el control de tráfico en ninguna de sus interfaces.

Un flujo de datos

Arranca `iperf` en modo servidor UDP en `pc3` y arranca `iperf` en modo cliente UDP en `pc1` para que envíe tráfico a 3 Mbit durante 10 segundos a `pc3`.

Observa en el lado servidor, el informe del tráfico recibido en el sentido `pc1` \rightarrow `pc3`.

Dos flujos de datos

- Arranca `iperf` en modo servidor UDP en `pc4`.
- Arranca otro `iperf` en modo servidor UDP en `pc3`.
- Inicia una captura de tráfico en la interfaz `eth1` de `r1`.
- Escribe (todavía sin ejecutar) el comando que arranca `iperf` en modo cliente UDP en `pc1` para que envíe 3 Mbit al servidor `pc3` en el sentido `pc1` \rightarrow `pc3` durante 10 segundos.
- Escribe (todavía sin ejecutar) el comando que arranca `iperf` en modo cliente UDP en `pc2` para que envíe 3 Mbit al servidor `pc4` en el sentido `pc2` \rightarrow `pc4` durante 10 segundos.
- Ejecuta los dos comandos anteriores uno a continuación de otro (lo más rápidamente que puedas) para que su ejecución se realice de forma simultánea.
- Interrumpe la captura aproximadamente 10 segundos después de que arrancaras `iperf`.

A continuación analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en `wireshark` y muestra cada uno de los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

Control de admisión para el tráfico de entrada

Vamos a configurar `r1` para restringir el tráfico de entrada para 2 flujos de datos que recibe `r1`:

- Flujo 1: origen 11.0.0.10 se va a restringir a una velocidad de 1Mbit y una cubeta de 10k.
- Flujo 2: origen 11.0.0.20 se va a restringir a una velocidad de 2Mbit y una cubeta de 10k.
- Utiliza `tc` para definir esta configuración en la interfaz `eth0` de `r1` que es la interfaz de entrada de `r1` para los flujos 1 y 2. Ten en cuenta que se aplique primero el filtro del flujo número 1 y después el del número 2. Guarda esta configuración en un fichero de *script*, por ejemplo con el nombre `tc-ingress.sh` que deberá contener las instrucciones que ejecutarías en la línea de comandos:

```
#!/bin/sh

# Esto es un comentario

echo "Borrando la disciplina de cola ingress en la interfaz eth0"
tc qdisc del dev eth0 ingress

echo "Creando la disciplina de cola ingress en la interfaz eth0"
tc qdisc add ...
...
```

Una vez creado el *script* debes darle permisos de ejecución con la orden:

```
chmod 755 tc-ingress.sh
```

Y por último, para ejecutarlo, debes escribir¹:

```
./tc-ingress.sh
```

- Inicia una captura de tráfico en la interfaz `eth1` de `r1`.
- Arranca dos clientes y 2 servidores tal y como lo hiciste en el apartado [sec:dosflujos].
- Interrumpe la captura aproximadamente 10 segundos después de que arrancaras `iperf`, cuando los servidores hayan terminado de recibir todo el tráfico.

A continuación analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en `wireshark` y muestra cada uno de los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

Disciplinas de colas para el tráfico de salida

Token Bucket Filter (TBF)

Mantén la configuración del tráfico de entrada en `r1` que has realizado en el apartado anterior en el *script* `tc-ingress.sh` .

- Define en `r1` para su interfaz `eth1` una disciplina TBF de salida con ancho de banda 1.5 Mbit, latencia 10 ms y tamaño de cubeta 10k y guarda la configuración en un nuevo *script* `tc-egress-tbf.sh` .
- Inicia una captura de tráfico en la interfaz `eth1` de `r1` .
- Arranca dos clientes y 2 servidores tal y como lo hiciste en el apartado [sec:dosflujos].
- Interrumpe la captura aproximadamente 10 segundos después de que arrancaras `iperf` .

A continuación analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en `wireshark` y muestra cada uno de los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

Modifica la configuración de TBF de salida para que ahora tenga una latencia de 20 segundos y realiza la misma prueba que antes ². Interrumpe la captura al menos cuando hayan pasado 20 segundos desde que comenzaste a enviar tráfico desde los clientes. A continuación analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en `wireshark` y muestra cada uno de los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

Prioridad (PRIO)

Mantén la configuración del tráfico de entrada en `r1` que has realizado en el apartado anterior en el *script* `tc-ingress.sh` . Borra la disciplina de cola de salida configurada en la interfaz `eth1` de `r1` .

La configuración TBF en el apartado [sec:tbf] permite gestionar el ancho de banda de salida para que no supere el valor configurado, en nuestro caso 1.5Mbit. Toma como punto de partida esta configuración para que ahora se atienda el tráfico de salida según diferentes

prioridades, configurando una disciplina de cola con prioridad que sea hija de la disciplina TBF.

- Escribe un *script* en `r1`, `tc-egress-prio.sh`, para configurar TBF con los siguientes parámetros: ancho de banda 1.5Mbit, cubeta 10k y latencia 20s. Crea una disciplina de cola hija con prioridad de tal forma que se asignen las siguientes prioridades:
 - Prioridad 1 (más prioritario): tráfico de la dirección IP origen 11.0.0.10
 - Prioridad 2 (prioridad intermedia): tráfico de la dirección IP origen 11.0.0.20
 - Prioridad 3 (menos prioritario): no lo vamos a definir.
- Inicia una captura de tráfico en la interfaz `eth1` de `r1`.
- Arranca 2 servidores para recibir los dos flujos de datos tal y como se hizo en el apartado [sec:dosflujos].
- Arranca dos clientes y 2 servidores tal y como lo hiciste en el apartado [sec:dosflujos].
- Interrumpe la captura aproximadamente 35 segundos después de que arrancaras `iperf`.

A continuación analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en wireshark y muestra cada uno de los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

Hierarchical Token Bucket (HTB)

Mantén la configuración del tráfico de entrada en `r1` que has realizado en el apartado anterior en el *script* `tc-ingress.sh`. Borra la disciplina de cola de salida configurada en la interfaz `eth1` de `r1`.

- Escribe un *script* en `r1`, `tc-egress-htb.sh`, para configurar en su interfaz `eth1` una disciplina HTB de salida con ancho de banda 1.2 Mbit. Reparte el ancho de banda de esta interfaz de salida de la siguiente forma:
 - 700 kbit para el tráfico con origen en `pc1`, `ceil` 700kbit.
 - 500 kbit para el tráfico con origen en `pc2`, `ceil` 500kbit.
- Inicia una captura de tráfico en la interfaz `eth1` de `r1`.
- Arranca 2 servidores para recibir los dos flujos de datos tal y como se hizo en el apartado [sec:dosflujos].

- Arranca dos clientes y 2 servidores tal y como lo hiciste en el apartado [sec:dosflujos].
- Interrumpe la captura aproximadamente 35 segundos después de que arrancaras `iperf`.

A continuación analiza los resultados obtenidos:

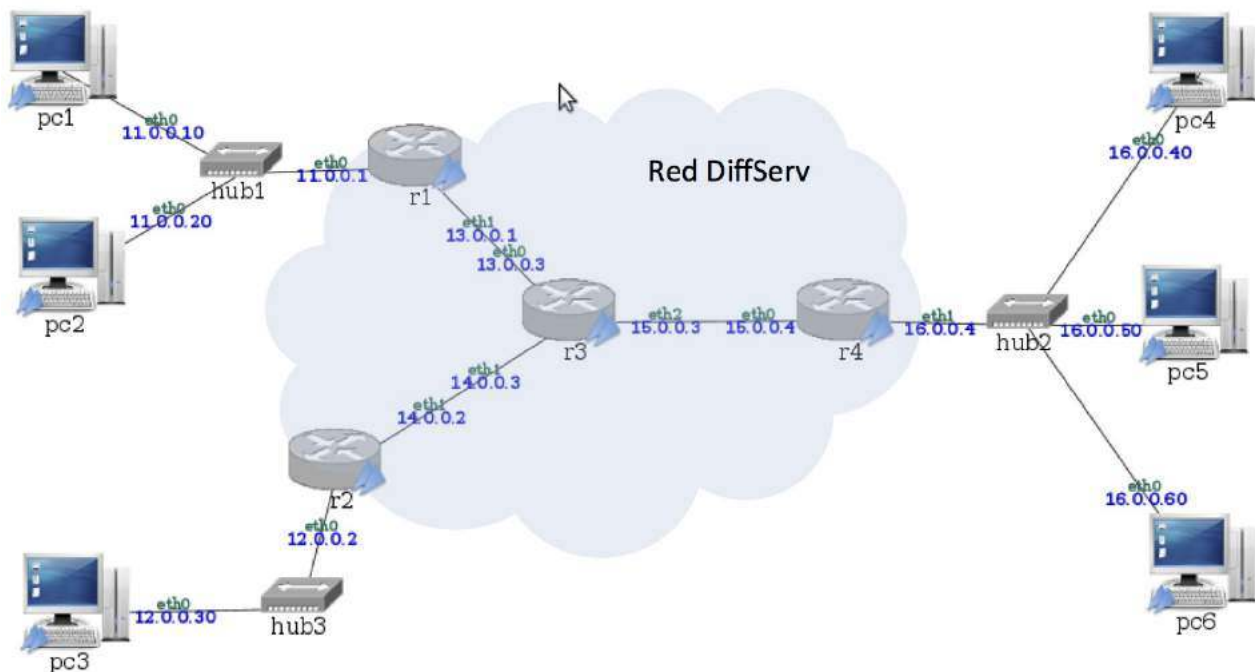
1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en wireshark y muestra cada uno de los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

Modifica la configuración de `ceil` en cada uno de los flujos para que puedan utilizar 1.2Mbit. Realiza la misma prueba que antes y analiza los resultados obtenidos:

1. Explica las estadísticas que muestran los servidores.
2. Carga la captura en wireshark y muestra cada uno de los flujos de forma gráfica. Explica el ancho de banda medido para cada uno de los flujos.

DiffServ

Descomprime el fichero que contiene el escenario de NetGUI lab-diffServ.tgz para realizar la práctica de diffServ en Linux.



Configuración de función policing y marcado de tráfico en DSCP

En el escenario de la figura [fig:diffServ] se va a configurar la red para que el tráfico desde `pc1` , `pc2` y `pc3` envíen paquetes a `pc4` , `pc5` y `pc6` atravesando una red diffServ. Se distinguirán 4 calidades diferentes con los códigos: EF, AF31, AF21 y AF11.

Utiliza la herramienta `tc` para garantizar que el tráfico que entra en `r1` cumple las siguientes características:

- La red diffServ deberá garantizar a la entrada los siguientes anchos de banda para `pc1` , descartando el tráfico sobrante:
 - Flujo 1: máximo 1.2mbit con ráfaga 10k para el tráfico dirigido a `pc4` , marcado con calidad EF. Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 2.
 - Flujo 2: máximo de 300kbit y ráfaga 10k, marcado con calidad AF31. Si se supera este ancho de banda, el tráfico será descartado definitivamente en `r1` .
- La red diffServ deberá garantizar a la entrada los siguientes anchos de banda para `pc2` , descartando el tráfico sobrante:
 - Flujo 3: máximo 600kbit con ráfaga 10k para el tráfico dirigido a `pc5` , marcado con calidad AF21. Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 4.
 - Flujo 4: máximo de 400kbit y ráfaga 10k, marcado con calidad AF11. Si se supera este ancho de banda, el tráfico será descartado definitivamente en `r1` .

Utiliza la herramienta `tc` para garantizar que el tráfico que entra en `r2` cumple las siguientes características:

- La red diffServ deberá garantizar a la entrada los siguientes anchos de banda para `pc3` , descartando el tráfico sobrante:
 - Flujo 5: máximo 400kbit con ráfaga 10k dirigido a `pc6` , marcado con calidad AF31. Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 6.
 - Flujo 6: máximo 300kbit con ráfaga 10k dirigido a `pc6` , marcado con calidad AF21. Si se supera este ancho de banda, el tráfico quedará clasificado dentro del flujo 7.
 - Flujo 7: máximo 100kbit con ráfaga 10k, marcado con calidad AF11. Si se supera este ancho de banda, el tráfico será descartado definitivamente en `r2` .

Realiza un *script* para `r1` y otro para `r2` donde se configuren estos perfiles de tráfico.

1. Ejecuta en tu escenario el envío "simultáneo" de:

- Desde el `pc1` 2Mbit a `pc4`
 - Desde el `pc2` 1.5Mbit a `pc5`
 - Desde el `pc3` 1Mbit a `pc6`
2. Realiza una captura en la subred 15.0.0.0/24 y comprueba que el resultado es el esperado:
- El tráfico que entra en la red diffServ es el que se ha especificado en el control de admisión.
 - El tráfico está marcado según las especificaciones anteriores.
3. Consulta las gráficas `I0 graphs` de Wireshark aplicando los filtros sobre las marcas DSCP de tal forma que se muestre cada calidad marcada de cada una de las fuentes:
- Tráfico de calidad EF
 - Tráfico de calidad AF31
 - Total
 - Con origen en `pc1` .
 - Con origen en `pc3` .
 - Tráfico de calidad AF21
 - Total
 - Con origen en `pc2` .
 - Con origen en `pc3` .
 - Tráfico de calidad AF11
 - Total
 - Con origen en `pc2` .
 - Con origen en `pc3` .

Explica los resultados obtenidos.

Tratamiento de tráfico en función del marcado DSCP

Mantén la configuración realizada en `r1` , `r2` .

Se establecen los siguientes parámetros de calidad dentro del router del núcleo diffServ (`r3`) para cada una de las calidades definidas. Configura HTB con ancho de banda 2.4Mbit para compartir entre todos los flujos con el siguiente patrón:

- Calidad EF: HTB 1mbit como mínimo y 1Mbit como máximo.
- Calidad AF31: HTB 500kbit como mínimo y 500kbit como máximo.
- Calidad AF21: HTB 600kbit como mínimo y 600kbit como máximo.
- Calidad AF11: HTB 200kbit como mínimo y 200kbit como máximo.

Realiza un *script* para `r3` donde se configure esta disciplina de cola según el marcado de los paquetes.

1. Ejecuta en tu escenario el envío "simultáneo" de:
 - Desde el `pc1` 2Mbit a `pc4`
 - Desde el `pc2` 1.5Mbit a `pc5`
 - Desde el `pc3` 1Mbit a `pc6`
2. Realiza una captura en la subred 15.0.0.0/24, espera al menos 1 minuto después de que el haya terminado de enviarse el tráfico de `pc1` , `pc2` y `pc3` y comprueba que el resultado es el esperado, es decir, el tráfico sigue el perfil indicado en las especificaciones anteriores.
3. Consulta las gráficas `IO graphs` de Wireshark aplicando los filtros sobre las marcas DSCP de tal forma que se muestre cada calidad marcada de cada una de las fuentes:
 - Tráfico de calidad EF
 - Tráfico de calidad AF31
 - Tráfico de calidad AF21
 - Tráfico de calidad AF11

Explica los resultados obtenidos y si crees que alguno de los flujos ha encolado tráfico para enviarlo posteriormente a los 10 segundos que dura la transmisión de `iperf` .

Modifica la configuración de HTB en `r3` para que si algún flujo no está utilizando el ancho de banda que tiene garantizado lo puedan usar el resto de flujos y vuelve a hacer la misma captura de tráfico en la subred 15.0.0.0/24. Explica los resultados obtenidos.

¹. Si es la primera vez que ejecutas este *script*, el *router* no ↩

tendrá configurada ninguna disciplina de cola en esa interfaz y al usar la instrucción de borrado se mostrará un error, pero la ejecución del `*script*` continuará y se aplicarán el resto de los cambios que hayas configurado.

2. Ten en cuenta que ahora el tráfico quedará en la cola de la [←](#)

disciplina TBF esperando a ser cursado según el ancho de banda que hemos configurado. El cliente terminará de enviar a los 10 segundos y esperará a recibir el informe del servidor. Sin embargo, el servidor no acabará de recibir (y por tanto no enviará el informe) hasta que TBF no termine de atender el tráfico de la cola de salida, que será más de 10 segundos. Al no recibir el cliente el informe del servidor, terminará imprimiendo un ``Warning``. De la misma forma cuando el servidor haya terminado de recibir y envíe el informe al cliente, éste ya habrá terminado su ejecución e imprimirá un mensaje indicando que no ha podido enviar el informe al cliente: ``Connection refused``.

VPN con IPsec

- [Introducción](#)
- [Cabecera AH](#)
- [Cabecera ESP](#)
- [Modos de funcionamiento en IPsec](#)
- [Asociación de seguridad](#)
- [Políticas de seguridad](#)
- [IKE](#)

Introducción

Una VPN (Virtual Private Network) conecta dos subredes seguras a través de una red insegura. IPsec encapsula los paquetes de una VPN en el nivel de red. Otros mecanismos para crear VPNs encapsulan los paquetes en el nivel de transporte o superior (SSL, TSL o SSH).

IPsec está integrado con la implementación de TCP/IP en el sistema operativo.

IPsec consiste en la implementación de 3 protocolos:

- **AH** (Authentication Header, código de protocolo=51): protocolo para la autenticación, integridad de datos y no repudio.
- **ESP** (Encapsulation Security Payload, código de protocolo=50): protocolo que proporciona confidencialidad y/o autenticación.
- **IKE** (Internet Key Exchange): protocolo para gestionar las claves. Durante la fase de negociación, IKE especifica el flujo de tráfico que hará uso de la conexión IPsec: protocolo, direcciones IP, ToS.

Con el protocolo AH no se puede autenticar el datagrama IP entero ya que algunos campos de la cabecera IP se modifican en el camino, como el TTL. Pero sí quedan dentro de la comprobación de integridad y autenticación las direcciones IP y el campo protocolo de la cabecera IP más externa.

Con el protocolo ESP también es posible autenticar de forma similar a AH pero la autenticación no protege los campos de cabecera más externa del datagrama IP que sí protegía AH. Además ESP cifra el contenido. Usando ESP es necesario que al menos un

algoritmo de autenticación o cifrado se seleccione. Si sólo se desea uno de ellos, el otro se puede desactivar usando algoritmo NULL.

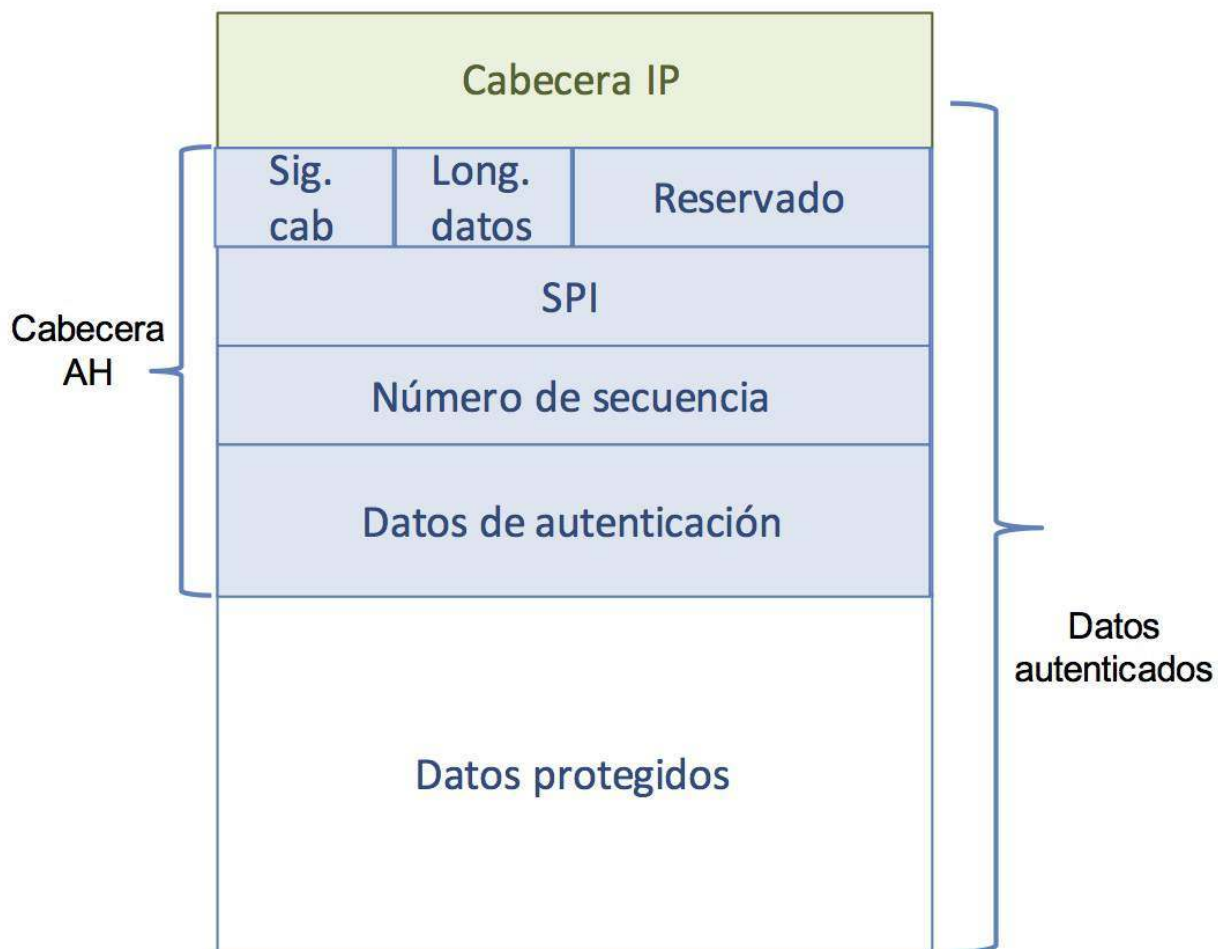
ESP puede hacer lo mismo que AH excepto autenticar los campos de la cabecera IP más externa, pero además proporciona confidencialidad. Por esta razón la mayoría de las VPNs usan ESP para proporcionar autenticación y cifrado.

Cabecera AH

En la siguiente figura se muestra el formato de un paquete encapsulado con AH. El paquete está formado por:

- Cabecera IP: en el campo protocolo se indicará el valor del código de la cabecera AH=51, que es lo que viene a continuación de la cabecera IP.
- Cabecera AH: campo siguiente cabecera, longitud de datos, SPI (Security Parameter Index), número de secuencia y datos de autenticación.
- Datos protegidos: dependiendo del modo en que se esté usando puede referirse a un datagrama IP completo o sólo a los datos de un datagrama IP (véase la sección [modos](#)).

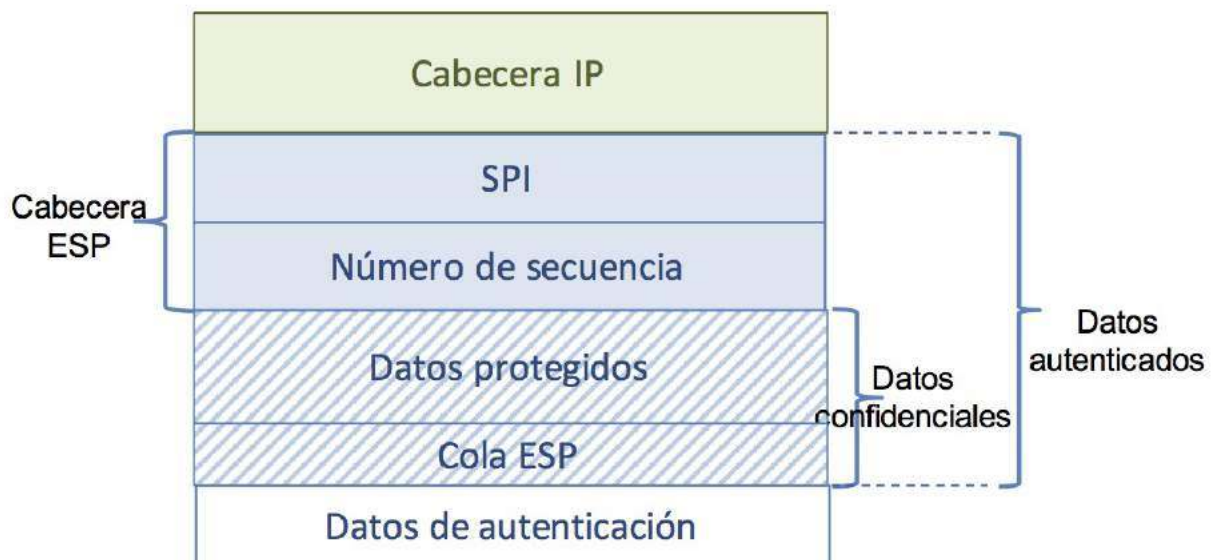
Donde la autenticación se realiza sobre los datos protegidos y parte de los campos de la cabecera IP externa.



Protocolo ESP

En la siguiente figura se muestra el formato de un paquete encapsulado con ESP. El paquete está formado por:

- Cabecera IP
- Cabecera ESP: SPI (Security Parameter Index) y número de secuencia
- Datos protegidos y cola ESP (estos campos son confidenciales, están cifrados).
Dependiendo del modo en que se esté usando los datos protegidos puede referirse a un datagrama IP completo o sólo a los datos de un datagrama IP (véase la sección [modos](#)).
- Datos de autenticación.

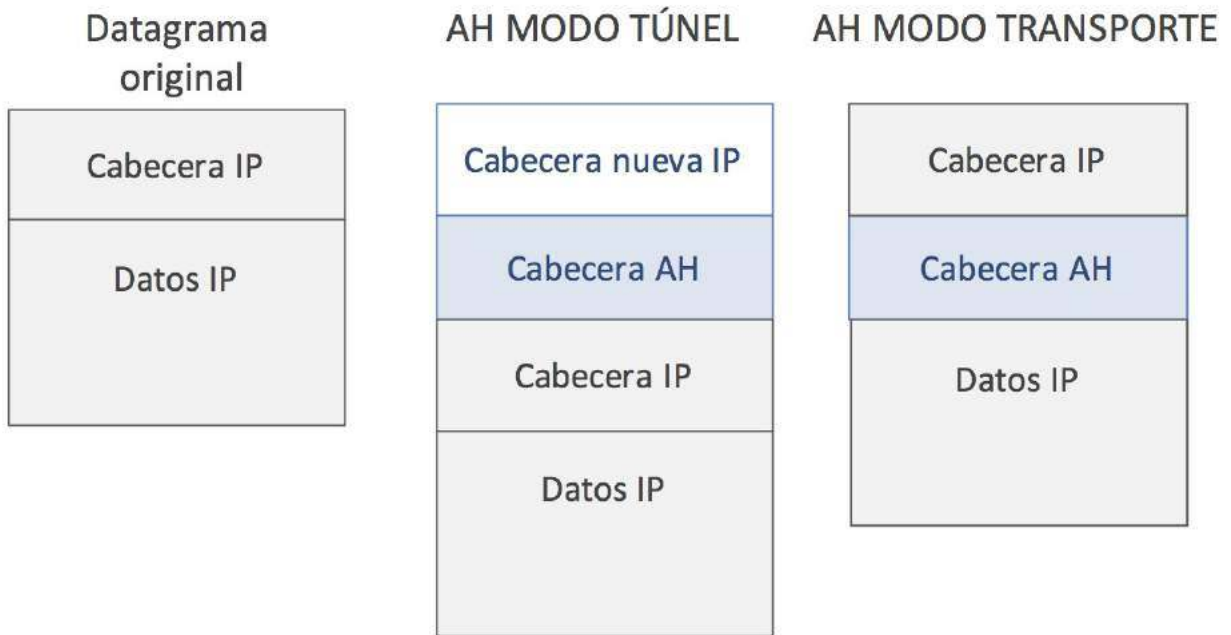


Modos de funcionamiento de IPsec

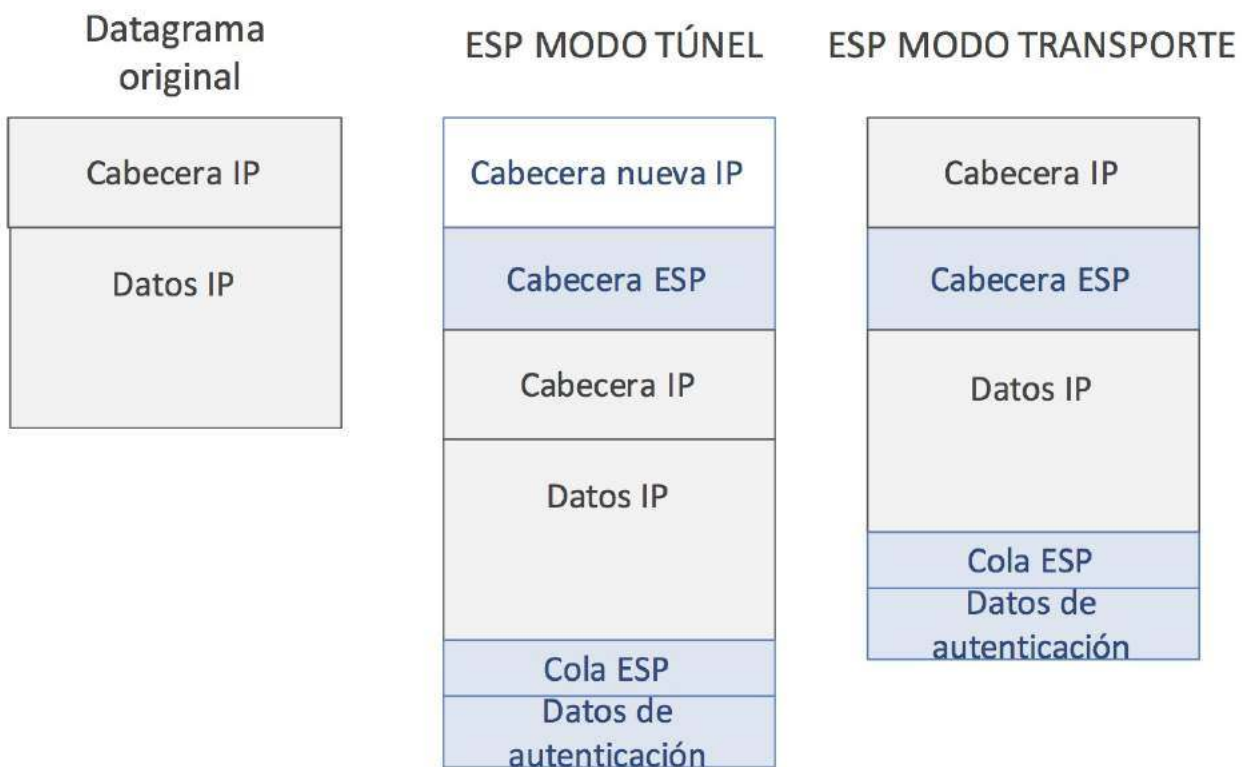
IPsec puede ser operado de 2 modos diferentes según se esté protegiendo la comunicación entre 2 máquinas, entre 2 subredes o entre máquina y subred.

- **Modo transporte:** Se usa entre 2 máquinas concretas, cuando éstas son los extremos finales de la VPN. Este modo protege los datos de nivel de transporte (para ser precisos los datos que vienen a continuación de la cabecera IP, que podrían ser ICMP). IPsec sólo protege la carga del datagrama, la cabecera original queda intacta y se inserta la cabecera IPsec entre la cabecera IP original y los datos de nivel superior. Los datos protegidos son la parte de datos del datagrama IP original.
- **Modo túnel:** Es más flexible pero necesita mayor ancho de banda. Se usa para conectar dos subredes o máquina y subred. Por ejemplo, conectar la subred de una oficina remota con la subred doméstica de un trabajador o conectar una máquina remota (*road warrior*, trabajador que cambia su localización y se conecta a la oficina) con la red doméstica. El datagrama IP se encapsula completamente en un nuevo datagrama IP con IPsec. Los datos protegidos son el datagrama IP original completo (cabecera y datos).

En la siguiente figura se pueden ver las diferencias del encapsulado AH en modo transporte y túnel.



En la siguiente figura se pueden ver las diferencias del encapsulado ESP en modo transporte y túnel.



Asociación de seguridad

Una **asociación de seguridad (SA, Security Association)** es un conjunto de atributos de seguridad compartidos entre los dos extremos que desean establecer una VPN para una comunicación unidireccional. Por tanto, una comunicación bidireccional a través de VPN tiene definidos 2 SAs, uno para cada sentido de la comunicación: uno para entrada y otro para salida. Una SA se distingue por la tripla:

- SPI (Security Parameter Index)
- dirección destino
- protocolo (AH/ESP)

El SPI se asigna en la máquina destino, por tanto, la tripla únicamente identifica una SA en una determinada máquina. Si una VPN está usando ambos protocolos AH y ESP, entonces deberá tener definidas 4 SAs, una para cada sentido y protocolo.

Las SAs pueden crearse manualmente, por el administrador del sistema, o negociarse por el protocolo de gestión de claves IPsec, IKE.

Las asociaciones de seguridad se almacenan en base de datos asociadas a la seguridad, SAD (Security Association Databases). La SAD se consulta en las siguientes situaciones:

- En el sentido salida, para enviar un datagrama que necesita utilizar ESP o AH. IPsec busca las SAs que le son aplicables (o crea una si no existe) para saber qué parámetros de seguridad tiene que utilizar, como el algoritmo de cifrado o autenticación, las claves, o el número de secuencia. IPsec utilizará estos parámetros para encapsular el datagrama.
- En el sentido entrada, para procesar un datagrama IPsec recibido. IPsec usa el SPI, dirección destino y protocolo para localizar SA en SAD y utilizar los parámetros de seguridad necesarios para descifrar o autenticar el datagrama. Si no se encuentra SA, el datagrama se descarta.

Políticas de seguridad

Una SA sólo define como IPsec protegerá el tráfico pero no especifica qué tráfico hay que proteger, para ello se definen **políticas de seguridad (SP, Security Policy)** que a su vez se almacenan en una base de datos de políticas de seguridad (SPD, Security Policy Databases). Una política de seguridad queda especificada por:

- Direcciones IP origen y destino del paquete a proteger. En modo transporte serán las mismas que las de SA, en modo túnel pueden ser diferentes.
- Protocolo de nivel de transporte.

- Puertos origen/destino.

Por tanto, dados dos extremos de una VPN, entre ellos puede haber definidas varias VPN para paquetes que cumplan diferentes requisitos.

La SDP se consulta en las siguientes situaciones:

- En el sentido salida, si se le aplica una política, dicha política tendrá asociada una SA. Si no existe SA, IPsec llamará a IKE para crear una SA.
- En el sentido entrada, después del descifrado y autenticación del datagrama gracias a que se ha localizado una SA, se consulta la SPD para asegurar que se ha llevado a cabo el procesado IPsec adecuado.

IKE

El IETF ha definido IKE (Internet Key Exchange) para la gestión automática de claves y el establecimiento de los SAs. IKE es un protocolo híbrido resultado de la integración de Oakley, Skeme y ISAKMP.

ISAKMP (RFC-2408) (Internet Security Association and Key Management Protocol): Este protocolo define una serie de fases para establecer una SA (Security Association), el establecimiento y mantenimiento de todas las claves necesarias para la familia de protocolos TCP/IP en modo seguro.

Se utiliza UDP con puerto origen y destino 500.

ISAKMP está dividido en 2 fases:

1. **Fase 1 (6 mensajes):** ambos nodos definen un canal seguro y autenticado. El canal seguro se consigue a través de un algoritmo de cifrado simétrico y un algoritmo HMAC. Las claves necesarias se derivan de una clave maestra que se obtiene mediante un algoritmo de intercambio de claves Diffie-Hellman. La autenticación se puede conseguir con:
 - Secreto compartido. Cada uno le demuestra al otro que conoce el secreto, enviando hash del secreto.
 - Certificados X509v3, integración de IPsec con PKI (Public Key Infrastructure). Los nodos tienen definido su certificado que les identifica de forma unívoca aportando su nombre en DNS o su dirección IP. La validación de los certificados se realiza a través de CRL (Certification Revocated List) que se almacena en el directorio de la PKI. Para solicitud y descarga de los certificados se utiliza el protocolo SCEP que intercambia mensajes PKCS a través de HTTP.

Los mensajes que intervienen en esta fase son los siguientes:

- negociación de algoritmos criptográficos (2 mensajes)
- intercambio de claves públicas (2 mensajes)
- autenticación mutua (2 mensajes)

Los métodos de autenticación que se permiten son los siguientes:

- Con firmas.
- Dos métodos de autenticación con cifrado de clave pública.
- Clave compartida previamente.

Vamos a ver el método de autenticación con firmas que se corresponde con el siguiente intercambio de mensajes:

Initiator		Responder
-----		-----
HDR, SA	-->	
	<--	HDR, SA
HDR, KE, Ni	-->	
	<--	HDR, KE, Nr
HDR*, IDii, [CERT,] SIG_I	-->	
	<--	HDR*, IDir, [CERT,] SIG_R

2. **Fase 2 (3 mensajes):** negociar parámetros de seguridad de la comunicación, en este caso IPsec (ESP, AH).

En el primer mensaje, SA es una negociación de la SA que puede incluir varias propuestas, la respuesta sólo debe tener una propuesta de SA. KE es la información pública de Diffie-Hellman.

Configuración de VPN con IPsec (ESP en modo túnel)

- [Introducción](#)
- [VPN entre subredes modo túnel](#)
 - [VPN en el lado servidor, r1](#)
 - [VPN en el lado cliente, r2](#)
 - [Comprobación del establecimiento de la VPN](#)
 - [SAD](#)
 - [SPD](#)
 - [Tráfico encapsulado en ESP](#)
 - [IKE](#)
 - [Fase 1: autenticación con firmas](#)
 - [Fase 2: Quick Mode](#)
- [Apéndice: creación de certificados](#)
 - [Certificado de la autoridad de certificación](#)
 - [Certificado del usuario en la máquina cliente de la VPN](#)
 - [Certificado de la máquina servidor de la VPN](#)
- [Apéndice: Captura de paquetes cifrados con tcpdump](#)

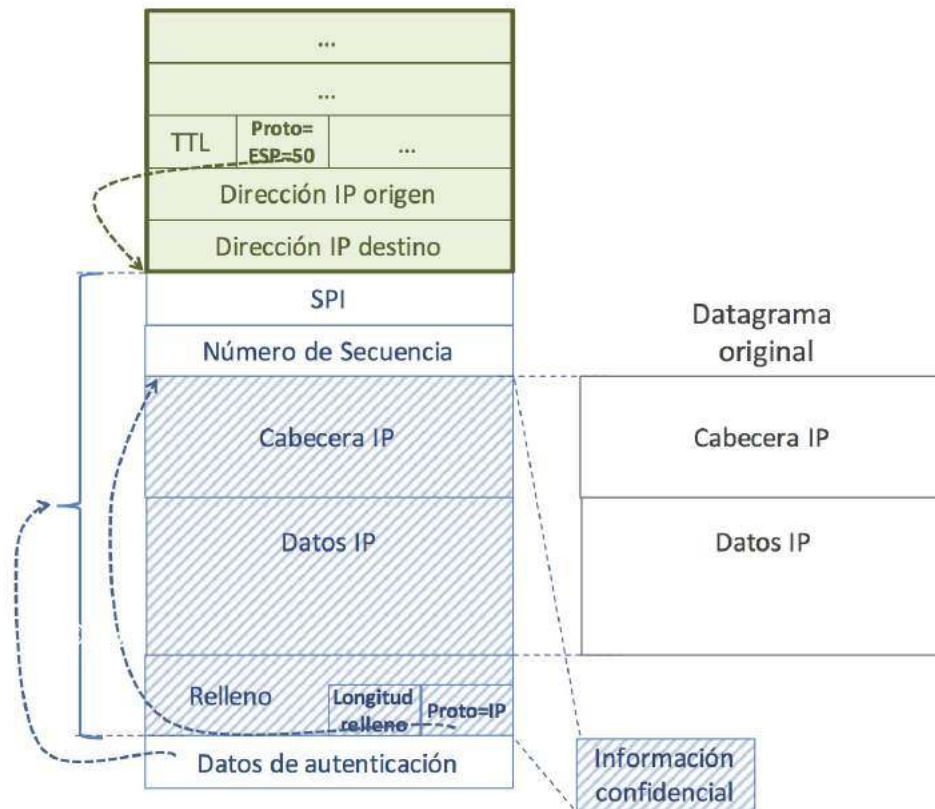
Introducción

Utilizaremos el software de `strongswan` para configurar una VPN.

`strongswan` es un demonio para la gestión automática de las claves. Se negociarán los SA entre dos extremos y las políticas de seguridad utilizando IKE. El tráfico IPsec lo gestiona la implementación de IPsec del sistema operativo.

Vamos a configurar ESP en modo túnel, utilizando AES para el cifrado y firmas como algoritmo para la autenticación (`RSA-SIG`). Para ello hay que crear los certificados X.509 necesarios para los dos extremos de la comunicación. La creación de certificados se encuentra en la [sección de certificados](#).

En la siguiente figura se muestra el detalle del paquete ESP en modo túnel, donde todo el paquete original queda cifrado dentro de la parte protegida. La cabecera ESP está formada por el SPI y el número de secuencia que permite detectar duplicados.



Procesar la salida de un paquete IPsec

Cuando un datagrama está preparado en la cola de salida, se comprueba si necesita ser procesado por IPsec. Si se necesita encapsulado ESP, su configuración dependerá de lo que haya en la SA, modo transporte o túnel.

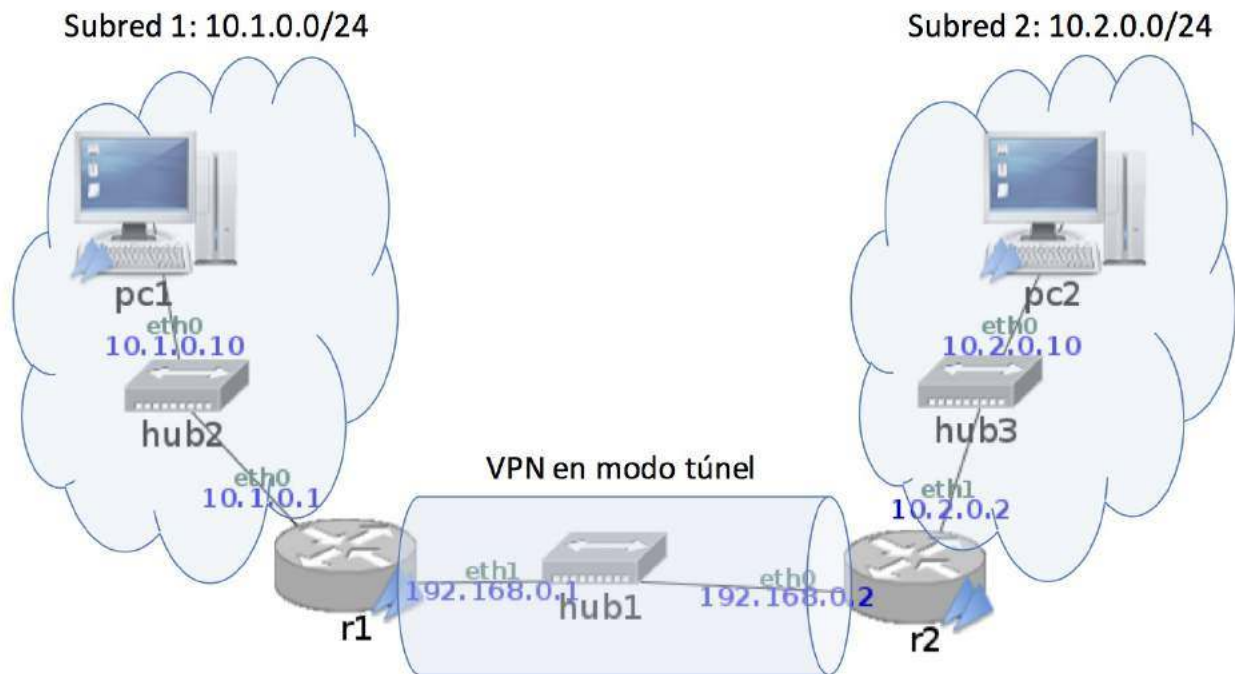
1. Se consulta SDP para saber si existe una política aplicable al paquete (se busca por direcciones IP origen y destino, protocolo, etc.) para determinar la SA. Si es necesario aplicar IPsec y no se encuentra SA, se negocia una SA.
2. Se incrementa el número de secuencia de la SA y se almacena en la cabecera ESP.
3. Se añade la cola ESP: relleno (si es necesario), longitud del relleno y el campo siguiente cabecera. Se cifran los datos y la cola ESP utilizando el algoritmo especificado en SA.
4. Se calcula una función de integridad de datos sobre la cabecera ESP, datos y cola ESP y se almacena en la parte de autenticación de datos. El algoritmo utilizado es el que está especificado en SA.

Procesar la entrada de un paquete IPsec

1. Tomando la dirección destino, protocolo (ESP/AH) y SPI se busca SA en SAD. Si no existe, **se tira el paquete**.
2. Si está activado el servicio de detección de duplicado, se comprueba que el número de secuencia cae dentro de la ventana de detección de duplicados.
3. El paquete es autenticado: se calcula la función de integridad especificada en SA sobre la cabecera ESP, datos y cola ESP y se comprueba que es igual a los datos de autenticación recibidos. Si es correcto, se actualiza la ventana de detección de duplicados.
4. Se descifra el paquete: datos y cola ESP utilizando el algoritmo especificado en SA. Se reconstruye el datagrama original a partir del paquete ESP.

VPN entre subredes, modo túnel{#vpn-tunnel)

En la siguiente figura se muestra el escenario donde se va a probar la VPN en modo túnel entre las subredes 10.1.0.0/24 y 10.2.0.0/24.



Utilizaremos para el cifrado AES128 y para la autenticación certificados. El algoritmo de hash será SHA256.

VPN en lado servidor (r1)

Configuración

Para ello editaremos los siguientes ficheros en `r1` :

- El fichero `/etc/ipsec.conf` La terminología `left` y `right` hace referencia a máquina local y máquina remota respectivamente. Como estamos configurando en `r1` :
 - `r1` será el lado `left` o local y `r2` será el lado `right` o remoto.
 - `leftsubnet` es la subred local que queremos conectar a través de la VPN `10.1.0.0/24`.
 - `rightsubnet` es la subred remota que queremos conectar a través de la VPN `10.2.0.0/24`.
 - `leftcert` es el certificado local de `r1` .
 - `rightcert` es el certificado remoto de `r2` .
 - `leftid` es la identidad a la cuál está asociada el certificado local, `r1` .
 - `rightid` es la identidad a la cuál está asociada el certificado remoto, `r2` .

```
r1:~# less /etc/ipsec.conf
# /etc/ipsec.conf - strongSwan IPsec configuration file

config setup
    # En esta sección se describen los parámetros generales de la comunicación

conn %default
    # Esta sección define parámetros que heredan todas las conexiones definidas po
steriormente
    ikelifetime=60m
    keylife=20m
    rekeymargin=3m
    keyingtries=1
    keyexchange=ikev1
    ike=aes128-sha256-modp3072!
    esp=aes128-sha256-modp3072!

conn net-net
    left=192.168.0.1
    leftcert=r1Cert.pem
    leftfirewall=yes
    leftid=@r1
    leftsubnet=10.1.0.0/24
    right=192.168.0.2
    rightid=@r2
    rightsubnet=10.2.0.0/24
    type=tunnel
    auto=add
```

La sección `conn` define las conexiones que utilizarán IPsec.

- El fichero `/etc/ipsec.secrets`

```
r1:~# less /etc/ipsec.secrets
# This file holds shared secrets or RSA private keys for inter-Pluto
# authentication. See ipsec_pluto(8) manpage, and HTML documentation.

# RSA private key for this host, authenticating it to any other host
# which knows the public part. Suitable public keys, for ipsec.conf, DNS,
# or configuration of other implementations, can be extracted conveniently
# with "ipsec showhostkey".

# this file is managed with debconf and will contain the automatically created pri
vate key
#include /var/lib/strongswan/ipsec.secrets.inc

: RSA r1Key.pem
```

- El fichero `hosts`

```
r1:~# less /etc/hosts
127.0.0.1 localhost localhost.localdomain
192.168.0.1 r1

::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Activar IPsec

Para activar IPsec:

```
r1:~# ipsec start
```

Para ver la claves cargadas en IPsec:

```
r1:~# ipsec secrets
002 forgetting secrets
002 loading secrets from "/etc/ipsec.secrets"
002  loaded private key from 'r1Key.pem'
```

Para comprobar el estado de IPsec:

```
root@r1:~# ipsec status
000 "net-net": 10.1.0.0/24===192.168.0.1[r1]...192.168.0.2[r2]===10.2.0.0/24;
                unrouted; eroute owner: #0
000 "net-net":  newest ISAKMP SA: #0; newest IPsec SA: #0;
000
Security Associations:
    none
```

Donde se indica que aún no hay SAs para la comunicación `net-net` donde está activado ISAKMP.

VPN en lado cliente (r2)

Configuración

Para ello editaremos los siguientes ficheros en `r2` :

- El fichero `/etc/ipsec.conf` La terminología `left` y `right` hace referencia a máquina local y máquina remota respectivamente. Como estamos configurando en `r2` :
 - `r2` será el lado `left` o local y `r1` será el lado `right` o remoto.
 - `leftsubnet` es la subred local que queremos conectar a través de la VPN `10.2.0.0/24`.
 - `rightsubnet` es la subred remota que queremos conectar a través de la VPN `10.1.0.0/24`.
 - `leftcert` es el certificado local de `r2` .
 - `rightcert` es el certificado remoto de `r1` .
 - `leftid` es la identidad a la cuál está asociada el certificado local, `r2` .
 - `rightid` es la identidad a la cuál está asociada el certificado remoto, `r1` .

```
r2:~# less /etc/ipsec.conf
# /etc/ipsec.conf - strongSwan IPsec configuration file

config setup

conn %default
    ikelifetime=60m
    keylife=20m
    rekeymargin=3m
    keyingtries=1
    keyexchange=ikev1
    ike=aes128-sha256-modp3072!
    esp=aes128-sha256-modp3072!

conn net-net
    left=192.168.0.2
    leftfirewall=yes
    leftcert=r2Cert.pem
    leftsubnet=10.2.0.0/24
    leftid=@r2
    right=192.168.0.1
    rightsubnet=10.1.0.0/24
    rightid=@r1
    auto=add
```

- El fichero `/etc/ipsec.secrets`

```
r2:~# less /etc/ipsec.secrets
# This file holds shared secrets or RSA private keys for inter-Pluto
# authentication. See ipsec_pluto(8) manpage, and HTML documentation.

# RSA private key for this host, authenticating it to any other host
# which knows the public part. Suitable public keys, for ipsec.conf, DNS,
# or configuration of other implementations, can be extracted conveniently
# with "ipsec showhostkey".

# this file is managed with debconf and will contain the automatically created pri
vate key
#include /var/lib/strongswan/ipsec.secrets.inc

: RSA r2Key.pem
```

- El fichero `hosts`

```
r2:~# less /etc/hosts
127.0.0.1 localhost localhost.localdomain
192.168.0.1 r1

::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

Activar IPsec y crear la VPN con el servidor

Para activar IPsec:

```
r2:~# ipsec start
```

Para ver la claves cargadas en IPsec:

```
r2:~# ipsec secrets
002 forgetting secrets
002 loading secrets from "/etc/ipsec.secrets"
002  loaded private key from 'r2Key.pem'
```

Para comprobar el estado de IPsec:

```
r2:~# ipsec status
000 "net-net": 10.2.0.0/24===192.168.0.2[r2]...192.168.0.1[r1]===10.1.0.0/24;
           unrouted; eroute owner: #0
000 "net-net":  newest ISAKMP SA: #0; newest IPsec SA: #0;
000
Security Associations:
  none
```

Para conectarse con el lado servidor de la VPN utilizando IPsec:

```
r2:~# ipsec up net-net
```

Comprobación del establecimiento de la VPN

Para comprobar el estado de IPsec en `r1` :

```
r1:~# ipsec status
000 "net-net": 10.1.0.0/24===192.168.0.1[r1]...192.168.0.2[r2]===10.2.0.0/24;
           erouted; eroute owner: #2
000 "net-net":  newest ISAKMP SA: #1; newest IPsec SA: #2;
000
000 #2: "net-net" STATE_QUICK_R2 (IPsec SA established);
           EVENT_SA_REPLACE in 988s; newest IPSEC; eroute owner
000 #2: "net-net" esp.ce835c0f@192.168.0.2 (840 bytes, 81s ago)
           esp.c8cf0f4f@192.168.0.1 (840 bytes, 81s ago); tunnel
000 #1: "net-net" STATE_MAIN_R3 (sent MR3, ISAKMP SA established);
           EVENT_SA_REPLACE in 3387s; newest ISAKMP
000
Security Associations:
  none
```

Donde se está utilizando ESP en modo túnel destino=192.168.0.2, SPI=ce835c0f y origen=192.168.0.1, SPI=c8cf0f4f.

Para comprobar el estado de IPsec en `r2` :

```

r2:~# ipsec status
000 "net-net": 10.2.0.0/24===192.168.0.2[r2]...192.168.0.1[r1]===10.1.0.0/24;
           erouted; eroute owner: #2
000 "net-net":  newest ISAKMP SA: #1; newest IPsec SA: #2;
000
000 #2: "net-net" STATE_QUICK_I2 (sent QI2, IPsec SA established);
           EVENT_SA_REPLACE in 584s; newest IPSEC; eroute owner
000 #2: "net-net" esp.c8cf0f4f@192.168.0.1 (840 bytes, 325s ago)
           esp.ce835c0f@192.168.0.2 (840 bytes, 325s ago); tunnel
000 #1: "net-net" STATE_MAIN_I4 (ISAKMP SA established);
           EVENT_SA_REPLACE in 2902s; newest ISAKMP
000 Security Associations:
      none

```

Donde se está utilizando ESP en modo túnel destino=192.168.0.1, SPI=cefc64a5 y origen=192.168.0.2, SPI=cc6d4e5e.

SAD

`xfrm` (*transform*) es un framework utilizado para manipular paquetes a la entrada y salida de una máquina. Gestiona SAD (Security Association Database) y la SPD (Security Policy Database).

Una vez establecida la comunicación entre los dos extremos del túnel, se puede consultar la información de SAs. Deben existir en cada extremo 2 SAs, uno para cada sentido de la comunicación.

En `r1` :

```

r1:~# ip xfrm state
src 192.168.0.1 dst 192.168.0.2
  proto esp spi 0xce835c0f reqid 16384 mode tunnel
  replay-window 32 flag af-unspec
  auth-trunc hmac(sha256) 0x972feeb4fa34659af67b1cb1a2f0988331b61c5f2f16961782ad
6202db5cd62e 128
  enc cbc(aes) 0xf024505f8f456a6a0f78a6cb8c5386b4
src 192.168.0.2 dst 192.168.0.1
  proto esp spi 0xc8cf0f4f reqid 16384 mode tunnel
  replay-window 32 flag af-unspec
  auth-trunc hmac(sha256) 0x91aa030b659b33a4d829b4aee2945f266a050a750323171e31fc
3d063544de52 128
  enc cbc(aes) 0x782ef28cdbbfd53e5272a1316af9eb6f

```

Donde SPI=0xce835c0f desde 192.168.0.1 a 192.168.0.2 y SPI=0xc8cf0f4f desde 192.168.0.2 a 192.168.0.1. El identificador `reqid=16384` debe ser igual al que se muestre en SP almacenada en SPD. Este identificador relaciona la SA con la SP.

En r2 :

```
r2:~# ip xfrm state
src 192.168.0.2 dst 192.168.0.1
    proto esp spi 0xc8cf0f4f reqid 16384 mode tunnel
    replay-window 32 flag af-unspec
    auth-trunc hmac(sha256) 0x91aa030b659b33a4d829b4aee2945f266a050a750323171e31fc
3d063544de52 128
    enc cbc(aes) 0x782ef28cddbffd53e5272a1316af9eb6f
src 192.168.0.1 dst 192.168.0.2
    proto esp spi 0xce835c0f reqid 16384 mode tunnel
    replay-window 32 flag af-unspec
    auth-trunc hmac(sha256) 0x972feeb4fa34659af67b1cb1a2f0988331b61c5f2f16961782ad
6202db5cd62e 128
    enc cbc(aes) 0xf024505f8f456a6a0f78a6cb8c5386b4
```

Donde SPI=0xc8cf0f4f desde 192.168.0.2 a 192.168.0.1 y SPI=0xce835c0f desde 192.168.0.1 a 192.168.0.2.

SPD

También se puede consultar en el lado del servidor la SPD (Security Policy Database).

En r1 :

```
r1:~# ip xfrm policy
src 10.1.0.0/24 dst 10.2.0.0/24
    dir out priority 1859 ptype main
    tmpl src 192.168.0.1 dst 192.168.0.2
        proto esp reqid 16384 mode tunnel
src 10.2.0.0/24 dst 10.1.0.0/24
    dir fwd priority 1859 ptype main
    tmpl src 192.168.0.2 dst 192.168.0.1
        proto esp reqid 16384 mode tunnel
src 10.2.0.0/24 dst 10.1.0.0/24
    dir in priority 1859 ptype main
    tmpl src 192.168.0.2 dst 192.168.0.1
        proto esp reqid 16384 mode tunnel
src ::/0 dst ::/0
    socket out priority 0 ptype main
src ::/0 dst ::/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
```

```

    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src ::/0 dst ::/0
    socket in priority 0 ptype main
src ::/0 dst ::/0
    socket out priority 0 ptype main
src ::/0 dst ::/0
    socket in priority 0 ptype main
src ::/0 dst ::/0
    socket out priority 0 ptype main
src ::/0 dst ::/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main

```

Donde se han definido 3 políticas:

- Los paquetes entrantes cifrados se descifran utilizando la SA (se hace búsqueda en SAD por dirección destino, SPI y protocolo=ESP) y después se comprueban las políticas: si van dirigidos a una dirección IP de `r1` se aplica política `in` (por ejemplo un paquete dirigido a la interfaz 10.1.0.1 del `r1` , si no van dirigidos a `r1` se aplica política `fwd` (por ejemplo un paquete dirigido a la interfaz 10.1.0.10, `pc1`).
- En los paquetes salientes primero se consulta la política `out` para ver si encaja alguna regla, en cuyo caso se cifra el paquete utilizando la información de SA.

En `r2` :

```

r2:~# ip xfrm policy
src 10.2.0.0/24 dst 10.1.0.0/24
    dir out priority 1859 ptype main
    tmpl src 192.168.0.2 dst 192.168.0.1
        proto esp reqid 16384 mode tunnel
src 10.1.0.0/24 dst 10.2.0.0/24
    dir fwd priority 1859 ptype main

```

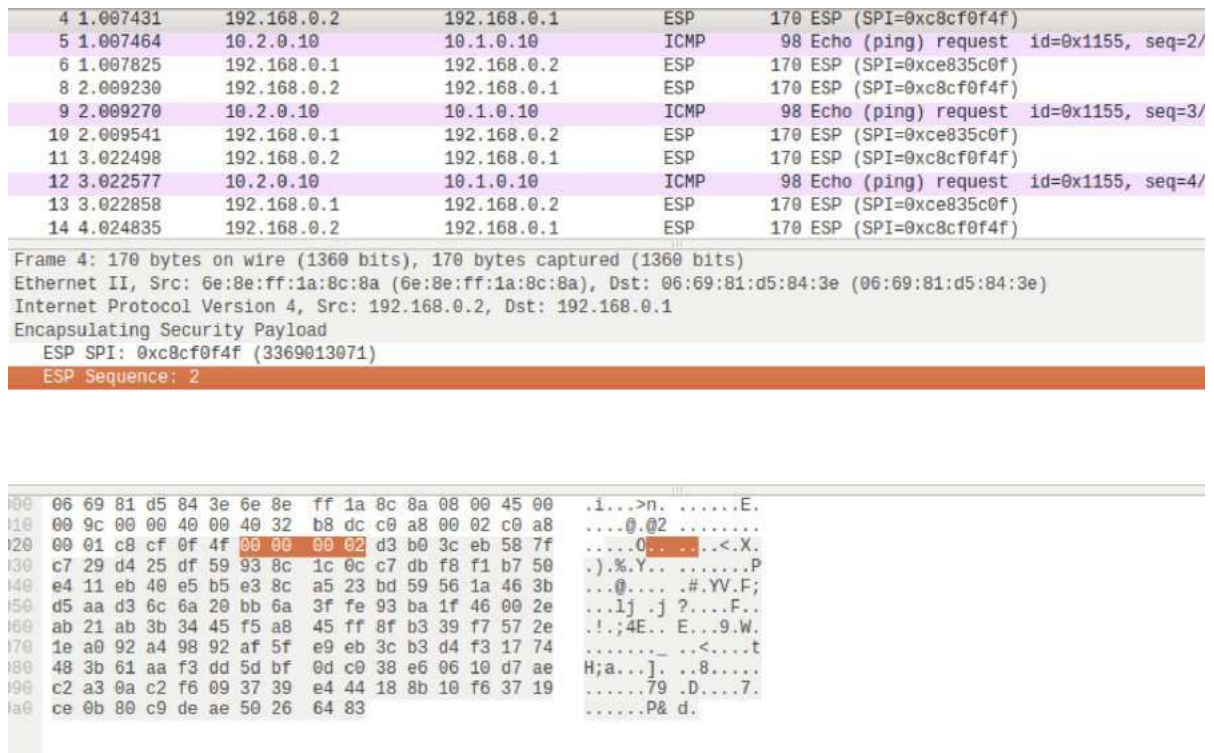
```
    tmpl src 192.168.0.1 dst 192.168.0.2
        proto esp reqid 16384 mode tunnel
src 10.1.0.0/24 dst 10.2.0.0/24
    dir in priority 1859 ptype main
    tmpl src 192.168.0.1 dst 192.168.0.2
        proto esp reqid 16384 mode tunnel
src ::/0 dst ::/0
    socket out priority 0 ptype main
src ::/0 dst ::/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket in priority 0 ptype main
src 0.0.0.0/0 dst 0.0.0.0/0
    socket out priority 0 ptype main
```

Tráfico encapsulado en ESP

Al ejecutar `ping` desde `pc2` hacia `pc1`, si capturamos el tráfico, éste muestra para el tráfico de entrada tanto el paquete cifrado como el descifrado y para el tráfico de salida sólo muestra el paquete cifrado.

- Tráfico capturado en `r1(eth1)`.

En la figura [fig:r1-eth1] se muestra la captura de tráfico en `r1(eth1)`.



Los paquetes 4 y 5 en realidad son el mismo mensaje ICMP echo request, siendo el paquete 4 el mensaje cifrado y el paquete 5 el mensaje descifrado:

- El paquete 4 muestra en su cabecera IP que el paquete tiene como origen `r2` y destino `r1`, los extremos del túnel ESP. Se puede ver que está usando `SPI=0xc8cf0f4f` que es la información que habíamos visto en esta SA. También se puede ver el número de secuencia, en este caso 2 (porque previamente se ha transmitido otro paquete, comienza en 1). Cada paquete enviado a través del túnel aumenta en uno el número de secuencia.
- El paquete 5 está descifrado y lleva la información del paquete original creado por `pc2`: dirección IP origen `pc2` y dirección IP destino `pc1`.

El paquete 6 es la respuesta ICMP echo reply cifrada, por ello, las direcciones IP que se muestran son las de los extremos del túnel cuyo `SPI=0xce835c0f` se corresponde con el que se ha seleccionado para la comunicación en ese sentido en la SA.

- Tráfico capturado en `r2(eth0)`.

En la figura [fig:r2-eth0] se muestra la captura de tráfico en `r2(eth0)`.

4	1.007491	192.168.0.2	192.168.0.1	ESP	170	ESP (SPI=0xc8cf0f4f)
5	1.007964	192.168.0.1	192.168.0.2	ESP	170	ESP (SPI=0xce835c0f)
6	1.007979	10.1.0.10	10.2.0.10	ICMP	98	Echo (ping) reply id
8	2.009189	192.168.0.2	192.168.0.1	ESP	170	ESP (SPI=0xc8cf0f4f)
9	2.010464	192.168.0.1	192.168.0.2	ESP	170	ESP (SPI=0xce835c0f)
10	2.010493	10.1.0.10	10.2.0.10	ICMP	98	Echo (ping) reply id
11	3.022357	192.168.0.2	192.168.0.1	ESP	170	ESP (SPI=0xc8cf0f4f)
12	3.023670	192.168.0.1	192.168.0.2	ESP	170	ESP (SPI=0xce835c0f)
13	3.023692	10.1.0.10	10.2.0.10	ICMP	98	Echo (ping) reply id
14	4.024857	192.168.0.2	192.168.0.1	ESP	170	ESP (SPI=0xc8cf0f4f)

Frame 5: 170 bytes on wire (1360 bits), 170 bytes captured (1360 bits)
Ethernet II, Src: 06:69:81:d5:84:3e (06:69:81:d5:84:3e), Dst: 6e:8e:ff:1a:8c:8a (6e:8e:ff:1a:8c:8a)
Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.2
Encapsulating Security Payload
ESP SPI: 0xce835c0f (3464715279)
ESP Sequence: 2

00	6e 8e ff 1a 8c 8a 06 69 81 d5 84 3e 08 00 45 00	n.....i ...>..E.
00	00 9c 1e 9b 00 00 40 32 da 41 c0 a8 00 01 c0 a8@2 .A.....
00	00 02 ce 83 5c 0f 00 00 00 02 33 27 5c 96 c2 4f\.. 3'\..0
00	d0 28 24 fb 17 5f 42 51 74 83 62 6d b9 81 d9 02	.(S..BQ t.bm....
00	56 1f b2 89 3f c4 25 df 43 af 11 22 6b 68 c7 15	V...?%. C.."kh..
00	b8 30 ed be 6a 54 27 ff 15 fb 8a 14 01 13 b3 a9	.0..jT'.
00	77 1e 27 c3 f2 65 f5 d3 81 e1 21 b0 72 9f 16 e2	w..'e.. !.r...
00	ab fa e0 de a4 97 ad 06 9d b4 af 9b 1e 2d 34 5f-4_
00	f0 ad b5 2f ff 54 c3 ef 87 48 ac 21 e0 f7 cb cc	.../..T...:H.!....
00	1e 3f cf d6 18 e1 fd 6e ee 72 45 52 c9 27 59 18	.?.....n .rER.'Y.
00	36 3e 91 08 b3 8c 87 4b 05 17	6>.....K ..

El paquete 4 es el mensaje ICMP echo request que se envía inicialmente desde `r2` cifrado (sólo muestra el tráfico cifrado en los paquetes de salida), usando SPI=0xc8cf0f4f.

Los paquetes 5 y 6 en realidad son el mismo mensaje ICMP echo reply, siendo el paquete 5 el mensaje cifrado y el paquete 6 el mensaje descifrado:

- El paquete 5 muestra en su cabecera IP que el paquete tiene como origen `r1` y destino `r2`, los extremos del túnel ESP. Se puede ver que está usando SPI=0xce835c0f que es la información que habíamos visto en esta SA. También se puede ver el número de secuencia, en este caso 2 (porque previamente se ha transmitido otro paquete, comienza en 1). Cada paquete enviado a través del túnel aumenta en uno el número de secuencia.
- El paquete 6 está descifrado y lleva la información del paquete original creado por `pc1`: dirección IP origen `pc1` y dirección IP destino `pc2`.

Como puede observarse en los paquetes que van encapsulados utilizando ESP, los campos visibles son SPI y el número de secuencia. El resto de los campos van cifrados y no pueden interpretarse. En la figura anterior puede observarse en la representación hexadecimal como después del campo número de secuencia (marcado en naranja) hay más campos cifrados en ese mismo mensaje.

Para conocer un poco mejor cómo funciona el encapsulado ESP se puede seleccionar como algoritmo de cifrado ESP=null, de esta forma, se utilizaría el encapsulado ESP pero los campos no viajarían cifrados y se podrían ver en la captura. Para realizar esta prueba, sólo es necesario añadir en el fichero `ipsec.conf` de ambos extremos de la VPN:

```
esp=null
```

En la figura [fig:icmp-req-null] se muestra el paquete ICMP echo request desde pc2 a pc1, encapsulado en ESP=null. Se observa las siguientes cabeceras:

- Cabecera externa que lleva la direcciones IP de los extremos del túnel: desde 192.168.0.2 a 192.168.0.1
- Cabecera ESP: SPI, número de secuencia.
- Datos protegidos ESP=null: paquete ICMP request con la cabecera original desde 10.2.0.10 a 10.1.0.10, relleno, siguiente cabecera (IPIP)
- Datos de autenticación.

```
▶ Ethernet II, Src: 6e:8e:ff:1a:8c:8a (6e:8e:ff:1a:8c:8a), Dst: 06:69:81:d5:84:3e (06:69:81:d5:84:3e)
▶ Internet Protocol Version 4, Src: 192.168.0.2, Dst: 192.168.0.1
▼ Encapsulating Security Payload
  ESP SPI: 0xc855d0b9 (3361067193)
  ESP Sequence: 1
  ESP Pad Length: 2
  Next header: IPIP (0x04)
  Authentication Data: 2e14eeefae2665710f878b98
▶ Internet Protocol Version 4, Src: 10.2.0.10, Dst: 10.1.0.10
▶ Internet Control Message Protocol
```

En la siguiente figura se muestra el paquete ICMP echo reply desde pc1 a pc2, encapsulado en ESP=null. Se observa las siguientes cabeceras:

- Cabecera externa que lleva la direcciones IP de los extremos del túnel: desde 192.168.0.1 a 192.168.0.2
- Cabecera ESP: SPI, número de secuencia.
- Datos protegidos ESP=null: paquete ICMP reply con la cabecera original desde 10.1.0.10 a 10.2.0.10, relleno, siguiente cabecera (IPIP)
- Datos de autenticación.

```

Frame 61: 172 bytes on wire (136 bytes captured)
▶ Ethernet II, Src: 06:69:81:d5:84:3e (06:69:81:d5:84:3e), Dst: 6e:8e:ff:1a:8c:8a (6e:8e:ff:1a:8c:8a)
▶ Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.2
▼ Encapsulating Security Payload
  ESP SPI: 0xc787010c (3347513612)
  ESP Sequence: 1
  ESP Pad Length: 2
  Next header: IPIP (0x04)
  Authentication Data: d40d100b16ed87b5978fe3d5
▶ Internet Protocol Version 4, Src: 10.1.0.10, Dst: 10.2.0.10
▶ Internet Control Message Protocol

```

IKE

El IETF ha definido IKE (Internet Key Exchange) para la gestión automática de claves y el establecimiento de los SAs. IKE es un protocolo híbrido resultado de la integración de Oakley, Skeme y ISAKMP.

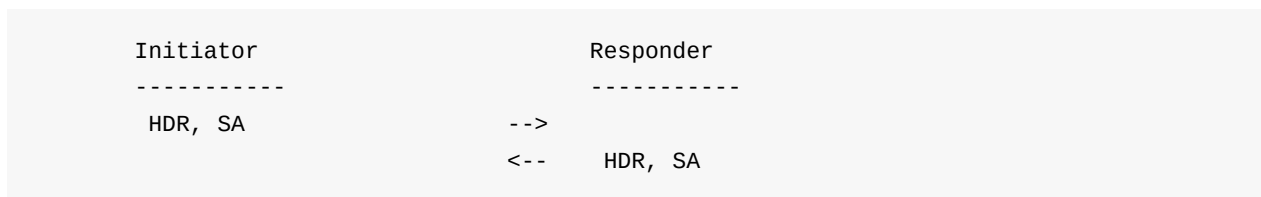
ISAKMP (RFC-2408) (Internet Security Association and Key Management Protocol): Este protocolo define una serie de fases para establecer una SA (Security Association), el establecimiento y mantenimiento de todas las claves necesarias para la familia de protocolos TCP/IP en modo seguro.

Se utiliza UDP con puerto origen y destino 500.

ISAKMP está dividido en 2 fases.

Si al activar el túnel en el lado cliente (`ipsec up net-net`), capturamos el tráfico intercambiado entre `r1` y `r2` se observarían los mensajes de las 2 fases.

Fase 1: autenticación con firmas



Donde HDR es la cabecera ISAKMP que indica el tipo de intercambio o modo (*Main Mode* o *Agressive Mode*).

En *Main Mode*, la fase 1 proporciona protección de la identidad de las dos partes. En *Agressive Mode* las partes no se intercambian la identidad y se reduce el número de mensajes que se intercambian, optimizando el tiempo de negociación entre ambas partes. Nótese que si se utiliza cifrado de clave pública para la autenticación en *Agressive Mode* también se proporcionará protección de identidad.

Negociación de parámetros de seguridad (2 mensajes)

1. Primer mensaje: Initiator -> Responder

El primer mensaje lo envía el `Initiator` y lleva la siguiente información:

- Initiator SPI (Security Parameter Index, 8 bytes)=a72d06230045d07f. Es la Cookie `CKY-I`.
- Responder SPI vacío
- Tipo de intercambio que se va a realizar (Main Mode, 2)
- Next Payload = SA.
- Type Payload: Security Association
- Type Payload: Vendor ID: strongSwan
- Type Payload: Vendor ID: XAUTH
- Type Payload: Vendor ID: RFC 3706 DPD (Dead Peer Detection)

```
▼ Internet Security Association and Key Management Protocol
  Initiator SPI: 60e7a2353a2fbcdb
  Responder SPI: 0000000000000000
  Next payload: Security Association (1)
  ▶ Version: 1.0
  Exchange type: Identity Protection (Main Mode) (2)
  ▶ Flags: 0x00
  Message ID: 0x00000000
  Length: 136
  ▶ Type Payload: Security Association (1)
  ▶ Type Payload: Vendor ID (13) : strongSwan
  ▶ Type Payload: Vendor ID (13) : XAUTH
  ▶ Type Payload: Vendor ID (13) : RFC 3706 DPD (Dead Peer Detection)
```

En la información SA se muestra la/s propuesta/s que se hace/n (en este caso sólo 1):

- IKEv1
- Duración 3600 seg
- Algoritmo de cifrado: AES-CBC
- Algoritmo Hash: SHA2-256 (si no existe negociación de PRF (pseudo random function) se utiliza la que esté definida en este apartado).
- Longitud de la clave 128
- Modo de autenticación: RSA-SIG

- Descripción de grupo 3072 bit MODP (Modular Exponential Diffie-Hellman groups): especifica el primo y el generador para crear el secreto compartido.

- primo p

$$p=2^{\{3072\}} - 2^{\{3008\}} - 1 + 2^{\{64\}} \times [[2^{\{2942\}} \times \pi] + 1690314]$$

- generador $g=2$

Cada lado de la comunicación generará un número a partir de un secreto individual:

- Extremo r_1 : a partir de x genera el número:

$$X=g^x \text{ mod } p$$

r_1 envía X a r_2 .

- Extremo r_2 : a partir de y genera el número:

$$Y=g^y \text{ mod } p$$

r_2 envía Y a r_1 .

- El secreto compartido por ambos y que entre ellos pueden calcular es el siguiente:

- r_1 calcula:

$$Y^x \text{ mod } p = g^{\{y^x\}} \text{ mod } p$$

- r_2 calcula:

$$X^y \text{ mod } p = g^{\{x^y\}} \text{ mod } p$$

Ambos números calculados por r_1 y r_2 son el mismo:

$$g^{\{xy\}} \text{ mod } p$$

y se denomina el secreto compartido de Diffie-Hellman.

```

▼ Type Payload: Security Association (1)
  Next payload: Vendor ID (13)
  Payload length: 56
  Domain of interpretation: IPSEC (1)
  ▶ Situation: 00000001
  ▼ Type Payload: Proposal (2) # 0
    Next payload: NONE / No Next Payload (0)
    Payload length: 44
    Proposal number: 0
    Protocol ID: ISAKMP (1)
    SPI Size: 0
    Proposal transforms: 1
    ▼ Type Payload: Transform (3) # 0
      Next payload: NONE / No Next Payload (0)
      Payload length: 36
      Transform number: 0
      Transform ID: KEY_IKE (1)
      ▶ Transform IKE Attribute Type (t=11,l=2) Life-Type : Seconds
      ▶ Transform IKE Attribute Type (t=12,l=2) Life-Duration : 3600
      ▶ Transform IKE Attribute Type (t=1,l=2) Encryption-Algorithm : AES-CBC
      ▶ Transform IKE Attribute Type (t=2,l=2) Hash-Algorithm : SHA2-256
      ▶ Transform IKE Attribute Type (t=14,l=2) Key-Length : 128
      ▶ Transform IKE Attribute Type (t=3,l=2) Authentication-Method : RSA-SIG
      ▶ Transform IKE Attribute Type (t=4,l=2) Group-Description : 3072 bit MODP group

```

2. Segundo mensaje: Initiator <- Responder

El segundo mensaje lo envía el `Responder` y es una respuesta al primer mensaje. Se ha generado el `Responder SPI` y además lleva la siguiente información:

- Initiator SPI=a72d06230045d07f. Es la Cookie `CKY-I`.
- Responder SPI =c8252a582ba06f1a. Es la Cookie `CKY-R`.
- Tipo de intercambio que se va a realizar (Main Mode, 2)
- Next Payload = SA
- Type Payload: Security Association
- Type Payload: Vendor ID: strongSwan
- Type Payload: Vendor ID: XAUTH
- Type Payload: Vendor ID: RFC 3706 DPD (Dead Peer Detection)

El SA lleva una de las propuestas realizadas por `Initiator`, en este caso, como sólo llevaba una propuesta, se incluye lo mismo que había enviado `Initiator`.

Intercambio de Diffie-Hellman y Nonce (2 mensajes)

Se envían los datos `KE` para el cálculo de la clave secreta compartida de Diffie-Hellman y el Nonce (número muy grande aleatorio).

Initiator	Responder
-----	-----
HDR, KE, Ni	-->
	<-- HDR, KE, Nr

1. **Tercer mensaje: Initiator -> Responder** El `Initiator` tiene que tener la clave pública de `Responder`. En el caso de que el `Responder` tenga varias claves, se incluirá en el mensajes un hash del certificado que el `Initiator` está usando para cifrar los datos. De esta forma, el `Responder` podrá saber con qué clave privada tendrá que descifrar los datos.

- Initiator SPI=a72d06230045d07f. Es la Cookie `CKY-I`.
- Responder SPI =c8252a582ba06f1a. Es la Cookie `CKY-R`.
- Tipo de intercambio que se va a realizar (Main Mode, 2)
- Next Payload = Key Exchange
- Type Payload: Key Exchange: $KE=g^x \text{ mod } p$
- Type Payload: Nonce de `Initiator` : `N_i`

2. **Cuarto mensaje: Initiator <- Responder**

- Initiator SPI=a72d06230045d07f. Es la Cookie `CKY-I`.
- Responder SPI =c8252a582ba06f1a. Es la Cookie `CKY-R`.
- Tipo de intercambio que se va a realizar (Main Mode, 2)
- Next Payload = Key Exchange
- Type Payload: Key Exchange: $KE=g^y \text{ mod } p$
- Type Payload: Nonce de `Responder` : `N_r`
- Type Payload: Certificate Request

Autenticación mutua (2 mensajes)

Los dos lados de la comunicación pueden calcular el secreto compartido Diffie-Hellman $g^{xy} \text{ mod } p$ que llamaremos g^{xy} .

Ahora se generan las claves que intervendrán en la comunicación. Usaremos la siguiente notación:

$\text{prf}(\text{key}, \text{msg})$: prf es la función pseudo-aleatoria con clave (keyed pseudo-random function), cuya clave se proporciona en el parámetro key , utilizada para generar una salida determinista del mensaje msg y que parece pseudo-aleatoria. prf se utilizan para el cálculo de claves y para la autenticación (por ejemplo keyed MAC).

Para la autenticación con firmas, se calcula la clave SKEYID aplicando prf a la clave Diffie-Hellman compartida utilizando como clave la concatenación de los Nonce:

$$\text{SKEYID} = \text{prf}(\text{Ni} \mid \text{Nr}, g^{\{xy\}})$$

El valor $\text{\$SKEYID}$ se utilizará como clave en prf para crear 3 claves para la autenticación y cifrado: SKEYID_d , SKEYID_a y SKEYID_e .

- Clave para el cálculo de claves de asociaciones de seguridad que no sean ISAKMP:

$$\text{SKEYID}_d = \text{prf}(\text{SKEYID}, g^{\{xy\}} \mid \text{CKY-I} \mid \text{CKY-R} \mid \emptyset)$$

- Clave de autenticación:

$$\text{SKEYID}_a = \text{prf}(\text{SKEYID}, \text{SKEYID}_d \mid g^{\{xy\}} \mid \text{CKY-I} \mid \text{CKY-R} \mid 1)$$

- Clave de cifrado:

$$\text{SKEYID}_e = \text{prf}(\text{SKEYID}, \text{SKEYID}_a \mid g^{\{xy\}} \mid \text{CKY-I} \mid \text{CKY-R} \mid 2)$$

Donde CKY-I y CKY-R son las cookies de Initiator y Responder respectivamente y que ambos extremos conocen porque se las están intercambiando en la fase 1.

Para autenticar el intercambio de mensajes es necesario calcular:

- $\text{HASH}_I = \text{prf}(\text{SKEYID}, g^{\{xi\}} \mid g^{\{xr\}} \mid \text{CKY-I} \mid \text{CKY-R} \mid \text{SA}_i \mid \text{ID}_{ii})$
- $\text{HASH}_R = \text{prf}(\text{SKEYID}, g^{\{xr\}} \mid g^{\{xi\}} \mid \text{CKY-R} \mid \text{CKY-I} \mid \text{SA}_i \mid \text{ID}_{ir})$

El intercambio de los mensajes 5 y 6 es el siguiente:

Initiator	Responder
-----	-----
HDR*, ID _{ii} , [CERT,] SIG _I -->	<-- HDR*, ID _{ir} , [CERT,] SIG _R

$\text{SIG}_I/\text{SIG}_R$ se calcula aplicando el algoritmo de firma negociado sobre $\text{HASH}_I / \text{HASH}_R$, es decir:

- `SIG_I` se consigue cifrando `HASH_I` con la clave privada de I.
- `SIG_R` se consigue cifrando `HASH_R` con la clave privada de R.
- **Quinto mensaje: Initiator -> Responder** El `Initiator` envía su identificación y hash del payload que permiten realizar la identificación y autenticación del `Initiator`.
 - Initiator SPI=a72d06230045d07f. Es la Cookie `CKY-I`.
 - Responder SPI =c8252a582ba06f1a. Es la Cookie `CKY-R`.
 - Tipo de intercambio que se va a realizar (Main Mode, 2)
 - Next Payload = Identification
 - Datos cifrados con `SKEYID_E`:
 - Identity Payload: identificación del `Initiator`, `IDi`, como el nombre del host o la dirección IP.
 - Signature Payload: `HASH_I` cifrado con la clave privada de I.
 - Certificate Data: certificado de I, que se identifica con el nombre de la máquina. Este certificado incluye la clave pública de I que se utiliza para extraer `HASH_I` del campo Signature Payload. Tanto el `Initiator` como el `Responder` deben tener un certificado generado por la misma CA (Certification Authority).
- **Sexto mensaje: Initiator <- Responder** El `Responder` envía su identificación y hash del payload que permiten realizar la identificación y autenticación del `Responder`.
 - Initiator SPI=a72d06230045d07f. Es la Cookie `CKY-I`.
 - Responder SPI =c8252a582ba06f1a. Es la Cookie `CKY-R`.
 - Tipo de intercambio que se va a realizar (Main Mode, 2)
 - Next Payload = Identification
 - Datos cifrados con `SKEYID_E`:
 - Identity Payload: identificación del `Initiator`, `IDir`, como el nombre del host o la dirección IP.
 - Signature Payload: `HASH_R` cifrado con la clave privada de R.
 - Certificate Data: certificado de R, que se identifica con el nombre de la máquina. Este certificado incluye la clave pública de R que se utiliza para extraer `HASH_R` del campo Signature Payload. Tanto el `Initiator` como el

`Responder` deben tener un certificado generado por la misma CA (Certification Authority).

Una vez intercambiado estos mensajes en cada extremo es necesario comprobar que el ID del otro lado está configurado por el administrador de red como un ID válido para establecer la comunicación. A continuación, cada extremo debe proceder a la autenticación, utilizando la clave pública (recibida en el certificado) para extraer `HASH_R/HASH_I`. Por tanto:

- El `Initiator` :
 - Descifra el mensaje usando `SKEYID_E`.
 - Extrae el certificado de R y de ahí extrae su clave pública.
 - Descifra `HASH_R` con la clave pública de R.
 - Calcula `HASH_R` con sus propios datos.
 - Si `HASH_R` calculada es igual a la obtenida, la autenticación es válida.
- El `Responder` :
 - Descifra el mensaje usando `SKEYID_E`.
 - Extrae el certificado de I y de ahí extrae su clave pública.
 - Descifra `HASH_I` con la clave pública de I.
 - Calcula `HASH_I` con sus propios datos.
 - Si `HASH_I` calculada es igual a la obtenida, la autenticación es válida.

Fase 2: Quick Mode

Se utiliza para calcular claves y negociar políticas para SA que no son ISAKMP. La información que viaja en los mensajes de esta fase tiene que estar cifrada con la clave calculada en fase 1, `SKEYID_e`.

1. **Primer mensaje: Initiator -> Responder**
2. **Segundo mensaje: Initiator <-Responder**
3. **Tercero mensaje: Initiator -> Responder**

Apéndice: creación de certificados

Los certificados deberán estar firmados por una autoridad de certificación (CA, certification authority). Por ello, crearemos primero el certificado de la CA y después los certificados de los extremos ¹

Certificado de la autoridad de certificación

Para crear una clave privada RSA de 4096 bits y un certificado autofirmado que será el certificado de la CA:

```
$ cd /etc/ipsec.d/
$ ipsec pki --gen --type rsa --size 4096 --outform pem \
  > private/strongswanKey.pem
$ chmod 600 private/strongswanKey.pem
$ ipsec pki --self --ca --lifetime 3650 \
  --in private/strongswanKey.pem --type rsa \
  --dn "C=ES, O=strongSwan, CN=strongSwan Root CA" \
  --outform pem \
  > cacerts/strongswanCert.pem
```

Donde `private/strongswanKey.pem` es la clave privada de la CA y

`cacerts/strongswanCert.pem` es el certificado de la CA con una validez de 10 años (3650 días). Los ficheros se almacenan en el formato de codificación PEM (Privacy Enhanced Mail), que está codificado en Base64, para incluirlo como texto ASCII en documentos, p ej. correo electrónico. Usado por OpenSSL y otras herramientas SSL.

El parámetro `--dn` (distinguished name) contiene los siguientes campos: país (C, country), organización (O, organization) y nombre común (CN, common name).

Como el formato PEM está codificado en base64 el contenido del fichero se mostrará de la siguiente forma:

```

$ less /etc/ipsec.d/cacerts/strongswanCert.pem
-----BEGIN CERTIFICATE-----
MIIF0TCCAYGgAwIBAgIIIIifkzk0SZQwwDQYJKoZIhvcNAQEFBQAwKjELMAkGA1UE
BhMCQ0gXGzAZBgNVBAOTEnN0cm9uZ1N3YW4gUm9vdCBDQTAEFw0xNjA1MjAxOTQz
MjZaFw0yNjA1MTgxOTQzMjZaMCoxCzAJBgNVBAYTAKNIMRswGQYDVQQKExJzdHJv
bmdTd2FuIFJvb3QgQ0EwggIiMA0GCSqGSIb3DQEBAQUAA4ICDwAwggIKAoICAQDC
3PeLz6x0AT94DYnnypSVRwpJjHzgCmyWlqVjMS4A4dd1mSR09a8QC11b/rmHp+N2
2L9AlKoNmXi9b2L0jWH5+x+LAoTRikae7pnm2B4ytu47QWEepopK4DoBjBGlw8FC
O/+EoUG5Uo0uQFunK20nQ7/D367qTOVy+mCeFI2hApINYOyWwqs+/Ca7BVQp4SEc
zbHx83wVqGpHU0csdkR+b+2pjASQ9BqFAYdf8Qpn5/5uh5frpw8QEDYsmmeMZVB3
TAKzu0IinQu5mRa0z1KP0Nmb31PxSw4RIBcE0i1cUCo4fIjkq024CAsXMHWDOshq
U+zu6IltqM6EIV0iysapyPHFVC0TdmhD0u2SLMAvvKy/CCZASEJoqPp35A+swAtr
YfZTqEA/Xq/y/JnP/qNW5DTK13VEXUuRyhypk3YKq6AuLk/YG8S6hw1J+daoIY/o
0LZiBzq+Ak60x1wFd4nI1wHrka9rXtc8x75aB28erb1MHU+ImEfh5V+FY57JQTT/
neS/1DFuTwCwi5sKse1l9jg2tIiHNWQ2TK4WwdvofGICRd6vtqeISMA5m+ES13n7
ODCwgJJBik+nny2plC9319sRFRleRYbxViLu+cchI/iXe9d7Quc9Pae1NT9vm4MW
RSnI5fRiMt03apJJCWBfi/XETV+tU05sFJsw8ddprwIDAQABo2MwYTAPBGNVHRMB
Af8EBTADAQH/MA4GA1UdDwEB/wQEAwIBBjAdBgNVHQ4EFgQUUGCRFi/UK2gW3gzmw
dh+WQewqUNMWhwYDVR0jBBgwFoAUGCRFi/UK2gW3gzmdh+WQewqUNMWDQYJKoZI
hvcNAQEFBQADggIBAHPf00KphH2NQ6/FzbiLvePa9oTyppRYLrTzoC4jHzpwt/6r
aIYP991FsgHyMqGK4VzbY+RtMWBgaHREMjoc+sy5FXUcJ8UH0nG9CmC4N8QWFamg
a1HAK8XZ5oLaMd0uMQBASNgynB4i3/QiQo0kEjKbEMhmCfhX7TmtaKCi0degdHV
zv6xTlkYI/j1/DCWWEZR/ck10L7plcE7MT1D/JyrEhtHhCSa817ksxEQS3Z7ezvE
x+tKqQhPe56Jym0jGXp3UPCIL110ggU+h3gLusSfsw15aJcUZdmIcVAg0ZjKUwmo
axfY0lhJhZ2/N04JkBsQSrgW0s5TzFsDZW8HY06o9Jh77j/QRK7x+pgLEdtIS0/G
mF4+9bjmEVQA562GG2qLRuiUPKozJrC0MD0+Yt5f+NyE6WgdjeEzaUChrKyq1Nh0
w0eSXZRkRXpnHnt1VuPtPjkUWCiaJaN0cTYCP6rD9dVgEEwN/wPbw8MivNI6PVo
Xo6H811lqYKDo0VRdbBWEy2hxkbME3Bm6LW5iGzdSaHvEloKhrAqTG09cQUuabdf
7U61mpC1BNP4x01sss9syCuavcX1zP8fkn15IFrYe+lohks0Ey681c3r1AcdmGh+
EztgE8pdKBSUooi92CarOt2SPN7FXJfB4q0UJjCvQLy2hufTptwcMHUjwVW
-----END CERTIFICATE-----

```

Para ver los campos del contenido del certificado:

```

$ ipsec pki --print --in /etc/ipsec.d/cacerts/strongswanCert.pem
cert:      X509
subject:   "C=ES, O=strongSwan Root CA"
issuer:    "C=ES, O=strongSwan Root CA"
validity:  not before May 20 21:43:26 2016, ok
           not after  May 18 21:43:26 2026, ok (expires in 3644 days)
serial:    22:27:e4:ce:4d:12:65:0c
flags:     CA CRLSign self-signed
authkeyId: 18:24:45:8b:f5:0a:da:05:b7:83:39:b0:76:1f:96:41:ec:2a:50:d3
subjkeyId: 18:24:45:8b:f5:0a:da:05:b7:83:39:b0:76:1f:96:41:ec:2a:50:d3
pubkey:    RSA 4096 bits
keyid:     ee:df:fd:15:86:7a:35:c9:8f:83:49:a1:15:3a:ff:70:85:54:d6:9a
subjkey:   18:24:45:8b:f5:0a:da:05:b7:83:39:b0:76:1f:96:41:ec:2a:50:d3

```

Certificado de la máquina servidor de la VPN

Se crea la clave privada `private/r1Key.pem` utilizando RSA de 2048 bits. Para crear su certificado firmado por la CA primero se extrae la clave pública asociada a clave privada anterior y se firma por la CA.

Es **muy importante** que en el campo `--dn` (distinguished name) se indique el FQDN de la máquina que funcionará como servidor de la VPN, para que el certificado identifique esa identidad. Este FQDN es el que utilizará el cliente de la VPN cuando se conecte al servidor. También se puede incluir ese FQDN en el parámetro `--san` (subject alternative name).

```
$ cd /etc/ipsec.d/
$ ipsec pki --gen --type rsa --size 2048 \
  --outform pem \
  > private/r1Key.pem
$ chmod 600 private/r1Key.pem
$ ipsec pki --pub --in private/r1Key.pem --type rsa | \
  ipsec pki --issue --lifetime 730 \
  --cacert cacerts/strongswanCert.pem \
  --cakey private/strongswanKey.pem \
  --dn "C=ES, O=strongSwan, CN=r1" \
  --san r1 \
  --flag serverAuth --flag ikeIntermediate \
  --outform pem > certs/r1Cert.pem
```

El certificado `r1Cert.pem` tiene una validez de 2 años (730 días).

```
$ ipsec pki --print --in /etc/ipsec.d/certs/r1Cert.pem
cert:      X509
subject:   "C=ES, O=strongSwan, CN=pc1"
issuer:    "C=ES, O=strongSwan Root CA"
validity:  not before May 21 10:18:26 2016, ok
           not after  May 21 10:18:26 2018, ok (expires in 725 days)
serial:    ec:f9:09:9b:36:72:b3:a5
altNames:  r1
flags:     serverAuth
authkeyId: 18:24:45:8b:f5:0a:da:05:b7:83:39:b0:76:1f:96:41:ec:2a:50:d3
subjkeyId: e5:72:0f:60:c9:68:0f:21:f9:de:83:eb:ad:5a:9d:ff:4a:98:99:84
pubkey:    RSA 2048 bits
keyid:     12:ba:41:a8:19:64:85:69:e2:3d:df:98:5c:86:2b:d0:72:3e:33:18
subjkey:   e5:72:0f:60:c9:68:0f:21:f9:de:83:eb:ad:5a:9d:ff:4a:98:99:84
```

Certificado del usuario en la máquina cliente de la VPN

Se crea la clave privada `private/user1Key.pem` utilizando RSA de 2048 bits. Para crear su certificado firmado por la CA `user1Cert.pem` primero se extrae la clave pública asociada a clave privada anterior y se firma por la CA.

El proceso es equivalente al del servidor, salvo que en este caso se utiliza la dirección de correo electrónico para identificar al cliente.

```
$ cd /etc/ipsec.d/
$ ipsec pki --gen --type rsa --size 2048 \
  --outform pem \
  > private/user1Key.pem
$ chmod 600 private/user1Key.pem
$ ipsec pki --pub --in private/pc1Key.pem --type rsa | \
  ipsec pki --issue --lifetime 730 \
  --cacert cacerts/strongswanCert.pem \
  --cakey private/strongswanKey.pem \
  --dn "C=ES, O=strongSwan, CN=user1@pc1" \
  --san user1@pc1 \
  --flag serverAuth --flag ikeIntermediate \
  --outform pem > certs/user1Cert.pem
```

El lado cliente necesita tener almacenada su clave privada y su certificado.

Para revocar un certificado, por ejemplo, si se desea cancelar un certificado de `user1` (`user1Cert.pem`):

```
$ cd /etc/ipsec.d/
$ ipsec pki --signcrl --reason key-compromise \
  --cacert cacerts/strongswanCert.pem \
  --cakey private/strongswanKey.pem \
  --cert certs/user1Cert.pem \
  --outform pem > crls/crl.pem
```

Se genera una lista de certificados revocados en `/etc/ipsec.d/crls/crl.pem`. Si alguien intenta autenticarse con dicho certificado revocado, debería obtener un error como el siguiente:

```
charon: 13[CFG] certificate was revoked
        on May 22 10:31:20 UTC 2016, reason: key compromise
```

Si se desea añadir otro certificado como revocado `user2Cert.pem`, habría que copiar el fichero actual previamente:

```
$ cd /etc/ipsec.d/
$ cp crls/crl.pem crl.pem.tmp
$ ipsec pki --signcrl --reason key-compromise \
  --cacert cacerts/strongswanCert.pem \
  --cakey private/strongswanKey.pem \
  --cert certs/user2Cert.pem \
  --lastcrl crl.pem.tmp \
  --outform pem > crls/crl.pem
$ rm crl.pem.tmp
```

Apéndice: Captura de paquetes cifrados con tcpdump

Si se utiliza `tcpdump` / `wireshark` para capturar tráfico en una determinada interfaz de red, se obtienen los paquetes cifrados y no cifrados sólo para el tráfico entrante en la máquina. El tráfico de salida se captura cifrado. El procesamiento de paquetes IPsec se realiza en el kernel. Para poder capturar el tráfico cifrado y descifrado entrante y saliente en una determinada interfaz se puede usar `nflog` dentro de `iptables`.

Las reglas `nflog` envían logs a un grupo de multicast interno del kernel, que es identificado por un número entero.

- Para almacenar el tráfico IPsec e IKE entrante

```
iptables -t filter -I INPUT -p esp -j NFLOG --nflog-group 5
iptables -t filter -I INPUT -p ah -j NFLOG --nflog-group 5
iptables -t filter -I INPUT -p udp -m multiport --dports 500,4500 -j NFLOG --nflog
-group 5
```

Donde `-m` utiliza el módulo `multiport` para poder usar la opción `--dports` con un conjunto de puertos destino.

- Para almacenar el tráfico IPsec e IKE saliente

```
iptables -t filter -I OUTPUT -p esp -j NFLOG --nflog-group 5
iptables -t filter -I OUTPUT -p ah -j NFLOG --nflog-group 5
iptables -t filter -I OUTPUT -p udp -m multiport --dports 500,4500 -j NFLOG --nflog
-group 5
```

- Para almacenar el tráfico IPsec descifrado.

```
iptables -t mangle -I PREROUTING -m policy --pol ipsec --dir in -j NFLOG --nflog-group 5
iptables -t mangle -I POSTROUTING -m policy --pol ipsec --dir out -j NFLOG --nflog-group 5
```

Donde `-m` utiliza el módulo `policy` para utilizar la opción `--pol ipsec` para indicar tráfico IPsec y `--dir in/--dir out` para tráfico entrante/saliente.

- Para almacenar el tráfico IPsec destinado a la máquina local (iptables INPUT chain)

```
iptables -t filter -I INPUT -m addrtype --dst-type LOCAL -m policy --pol ipsec --dir in \
-j NFLOG --nflog-group 5
```

Donde `-m` utiliza el módulo `addrtype` para utilizar la opción `--dst-type LOCAL` que indica que la dirección destino del paquete es una dirección configurada localmente en la máquina (no es una dirección de multicast, broadcast, etc).

- Para almacenar el tráfico IPsec que hay que reenviar (iptables FORWARD chain)

```
iptables -t filter -I INPUT -m addrtype ! --dst-type LOCAL -m policy --pol ipsec -dir in \
-j NFLOG --nflog-group 5
```

- Para almacenar el tráfico IPsec de salida (iptables OUTPUT chain)

```
iptables -t filter -I OUTPUT -m policy --pol ipsec --dir out -j NFLOG --nflog-group 5
```

Se captura el tráfico de la siguiente manera:

```
r1:~# tcpdump -s 0 -n -i nflog:5
```

1. Información extraída: [↩](#)

<https://www.zeitgeist.se/2013/11/22/strongswan-howto-create-your-own-vpn/>