

REDES: Tema 4. Capa de red

(Grado en Ingeniería Informática)

Copyright ©2004-2011  
Francisco Argüello Pedreira y José Carlos Cabaleiro Domínguez  
Departamento de Electrónica y Computación  
Universidad de Santiago de Compostela

18 de octubre de 2011



# Índice general

<b>4. Capa de red</b>	<b>1</b>
4.1. Introducción . . . . .	1
4.2. Algoritmos de rutado . . . . .	2
4.2.1. Algoritmo de estado de los enlaces (EE) . . . . .	3
4.2.2. Algoritmo de vector de distancias (VD) . . . . .	6
4.2.3. Rutado jerárquico . . . . .	9
4.3. Encaminamiento en Internet . . . . .	9
4.3.1. RIP (Protocolo de Información de Rutado) . . . . .	10
4.3.2. OSPF (Primero el Camino más Corto) . . . . .	11
4.3.3. BGP (Protocolo de Pasarela de Frontera) . . . . .	12
4.4. El protocolo Internet (IP) . . . . .	13
4.4.1. Direccionamiento IPv4 . . . . .	13
4.4.2. Formato del datagrama . . . . .	16
4.4.3. IPv6 . . . . .	18
4.4.4. ICMP, Protocolo Mensajes de Control de Internet . . . . .	20
4.4.5. DHCP . . . . .	21
4.4.6. NAT, Traducción de direcciones de red . . . . .	21

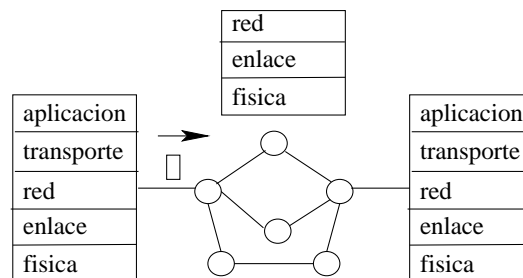


# Capítulo 4

## Capa de red

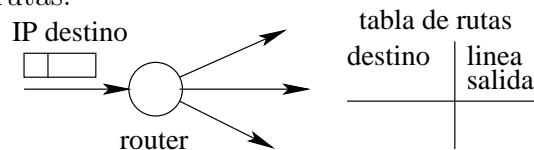
### 4.1. Introducción

La capa de red es la encargada de llevar los paquetes que le da la capa de transporte del host origen al host destino. Está implementado tanto en los hosts como en los routers.



La capa de red en Internet se denomina IP (Internet Protocol) y tiene las siguientes características:

- *Tipo datagrama.* Hay dos tipos de capa de red, de datagramas y de circuitos virtuales. En las redes de datagramas los routers encaminan los paquetes en función de la dirección destino que tienen los paquetes. Para ello consultan sus tablas de rutas.



- *Sin estado.* Los routers de Internet no guardan ninguna información de estado, cada paquete es tratado independientemente sin relación con los paquetes previos.
- *Red no fiable (mejor servicio posible, best effort).* Esto significa que no se garantiza que los paquetes lleguen al destino. Incluso los routers cuando están saturados eliminan paquetes sin contemplaciones.

Obviamente tampoco se garantizan tiempos de llegada, ni que los paquetes lleguen en orden. Los paquetes pueden llegar desordenados, por ejemplo, si siguen rutas distintas.

Es una buena solución: se desacoplan las funciones de rutado de los paquetes con las de fiabilidad. Se diseña la red para que pueda encaminar el mayor número de paquetes posible, aunque algunos se pierdan. Si después alguna aplicación necesita fiabilidad, en la capa de transporte se usa TCP, en cambio, si no la necesita, entonces usa UDP.

- *Facilidad de interconexión.* Otra ventaja del modelo que usa TCP/IP es que al ser tan simple la capa de red, hace muy fácil interconectar redes que emplean tecnologías de capa de enlace muy diferentes, por ejemplo, Ethernet, redes inalámbricas, conexiones telefónicas, etc. Por ello IP significa Protocolo Inter-red (o Internet).

## 4.2. Algoritmos de rutado

- El *algoritmo de rutado* es el encargado de encontrar el camino (o ruta) mínimo entre el origen y el destino. Habitualmente cada host está conectado directamente a un router, denominado *router por defecto*. Cuando el host emite un paquete se limita a entregárselo a ese router. Por tanto el problema se reduce a encontrar los caminos mínimos entre routers.
- El rutado es equivalente al problema matemático de encontrar los caminos mínimos en un grafo. Los routers son nodos y los enlaces son los arcos (o aristas). A los enlaces se les asigna un coste (o peso) que puede reflejar la distancia del enlace, su velocidad, lo cargado que esté, o el coste económico de dicho enlace, lo que convenga en cada caso. La longitud de la ruta es la suma de los costes de los arcos que forman la ruta.

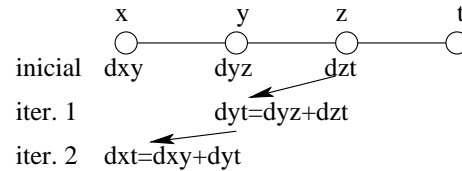
Clasificación de los algoritmos de rutado:

1. *Globales (de estado de los enlaces, EE).* En estos algoritmos se dispone previamente de toda la información de la red: su topología y el coste de todos los enlaces. Esto precisa de técnicas para obtener esta información, ya que a diferencia de los mapas de carreteras no se parte de un mapa de Internet. Una vez que disponemos de esta información, cada nodo puede computar por sí solo su tabla de rutas.

Se denomina de estado de los enlaces porque los nodos conocen el estado de todos los enlaces.

2. *Descentralizados (de vector de distancias, VD).* El cálculo de los caminos mínimos se hace mediante colaboración de todos los nodos. Los nodos

envían mensajes a sus vecinos (y sólo a sus vecinos) a los comunican la información de distancias que disponen. Inicialmente cada nodo sólo conoce su distancia a los nodos vecinos, pero después de una serie de iteraciones, comienzan a conocer distancias a nodos cada vez más alejados.



Se denomina vector de distancias porque en ningún momento los nodos conocen la topología de la red. Sólo conocen las distancias a los demás nodos y el enlace por el que debe comenzarse.

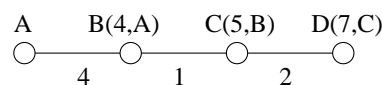
3. *Estáticos o dinámicos*. Son estáticos si sólo cambian cuando cambia la topología de la red (nuevos nodos o enlaces) o cuando se modifican manualmente ciertos parámetros. Los dinámicos se ejecutan periódicamente y de forma automática. Los usados actualmente en Internet son dinámicos.
4. *Sensibles a la carga o insensibles*. Un algoritmo de rutado es sensible a la carga, si el coste de los enlaces varía dinámicamente para reflejar el nivel de tráfico (o congestión) de dicho enlace. Los algoritmos actuales de rutado en Internet son insensibles a la carga.

**Oscilaciones.** Si el algoritmo de rutado es sensible a la carga, los caminos mínimos pueden cambiar periódicamente de una evaluación a la siguiente. Si se asocia un coste elevado a un enlace muy congestionado dejará de usarse en las siguientes transmisiones. En la evaluación siguiente, este enlace tendrá un coste muy bajo y se usará extensivamente hasta que quede congestionado. Y así sucesivamente. Primero resulta mínimo el camino 1, después el camino 2, después el camino 1, y a así sucesivamente. Además, si los caminos 1 y 2 tienen sentidos opuestos en alguna parte de la ruta, un paquete puede quedar atrapado en un ciclo y no llegar nunca a su destino.

#### 4.2.1. Algoritmo de estado de los enlaces (EE)

Son algoritmos globales en los que cada nodo calcula los caminos teniendo un conocimiento completo de la red, es decir, conoce todos los nodos y el coste de todos los enlaces. Veamos *el algoritmo de Dijkstra (1959)*.

**Algoritmo de Dijkstra (1959).** Los nodos se etiquetan con dos datos: la distancia del camino más corto conocido y el nodo previo. El nodo previo permite reconstruir la ruta.



Los nodos pueden ser provisionales o permanentes. Un nodo provisional puede conocer un camino, pero ese camino no es necesariamente el más corto. En cambio, para los nodos permanentes se conoce el camino mínimo.

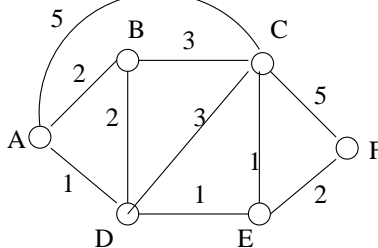
Después en una serie de pasos se encuentran caminos mejores para los nodos provisionales y las etiquetas se actualizan. Cuando se encuentra el camino definitivo las etiquetas se hacen permanentes (nodos rellenos).

○ provisional      ○ C(5,B)  
● permanente      longitud del camino, nodo previo

- El algoritmo consta de una serie de iteraciones, en cada una de las cuales se trabaja con un determinado nodo.
- En cada iteración se actualizan las etiquetas a los nodos vecinos al nodo de trabajo.
- A continuación se elige como siguiente nodo de trabajo el nodo de etiqueta provisional que tenga la distancia más corta.

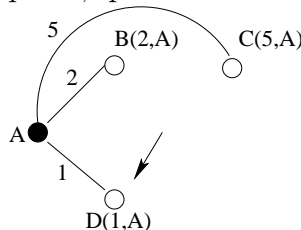
```
nodo_trabajo, nodo_permanente = inicial
for i=1, ..., numero_nodos
{  actualizar las etiquetas de los nodos vecinos al nodo_trabajo
   elegir el nodo de menor distancia y hacerlo nodo_trabajo, nodo_permanente }
```

**Ejemplo.** Supongamos que para la siguiente red deseamos conocer las distancias más cortas desde el nodo A a cada uno de los demás nodos:



1. El procedimiento se inicia haciendo permanente el nodo A (nodo relleno). A continuación, se determinan las distancias desde el nodo A hasta los nodos vecinos (B, C y D), reetiquetando cada uno con su distancia desde A. En la etiqueta también se incluye el nodo desde el cual se obtuvo la distancia (el nodo A), para poder después reconstruir el camino completo.

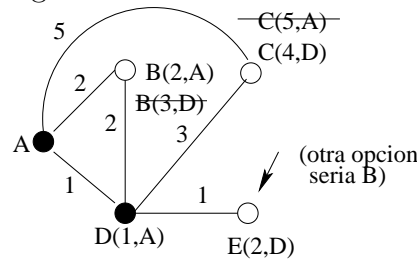
Después se exploran todos los nodos del grafo y se hace permanente el nodo con la etiqueta más pequeña, que en ese caso es el nodo D.





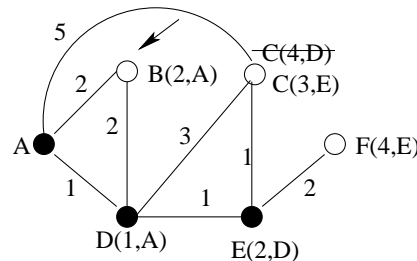
2. A continuación se parte desde el nodo D. Se examinan los nodos vecinos no permanentes (B, C y E). Si en alguno de estos nodos, la suma de la etiqueta de D y el coste del arco desde D al nodo considerado es menor que la etiqueta de ese nodo, tenemos entonces un camino más corto, por lo que el nodo se reetiqueta. En la figura ésto lo hacemos con los nodos B y C.

Después el grafo entero se revisa y se hace permanente el nodo con la etiqueta más pequeña. En este caso tenemos dos posibilidades de elección, los nodos B y E. Elegimos aleatoriamente el nodo E.

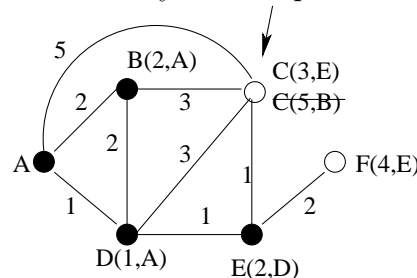


3. El nodo E se convierte ahora en el nodo de trabajo. Se examinan los nodos vecinos a E y no permanentes (C y F), calculando una nueva distancia desde E. Si es menor que la que indica la etiqueta, se actualiza. Esto sucede con el nodo C.

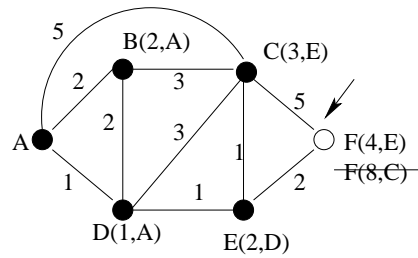
Se examina el grafo completo y se hace permanente el de la etiqueta más pequeña, el nodo B.



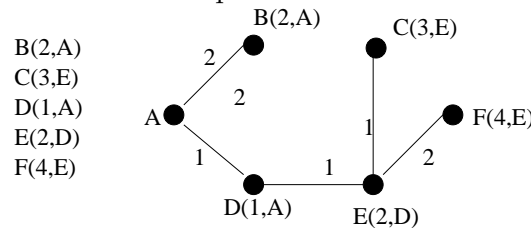
4. Se parte de B, se actualiza C y se hace permanente C.



5. Se parte de C, se actualiza F y se hace permanente F. Ya están hechos permanentes todos los nodos, por lo que ya hemos acabado con esta fase del algoritmo.



6. El camino desde A hasta un determinado nodo se determina empezando en este último nodo y reconstruyendo el camino hacia atrás teniendo en cuenta los nodos indicados en las etiquetas.



7. Obtenemos la siguiente tabla de rutas para el nodo A:

destino	salida
B	B
D	D
C	D
E	D
F	D

**Complejidad computacional del algoritmo Dijkstra.** El algoritmo consta de dos lazos: el primero recorre los  $n$  nodos y el segundo para todo nodo  $i$ -ésimo inspecciona los  $n - i$  nodos no permanentes para ver cual tiene la etiqueta menor. Haciendo las cuentas,  $(n - 1) + (n - 2) + \dots + 1$ , resulta un total  $n(n - 1)/2$  iteraciones. Por tanto, el algoritmo Dijkstra tiene complejidad cuadrática,  $O(n^2)$ .

#### 4.2.2. Algoritmo de vector de distancias (VD)

Es un algoritmo descentralizado pues todos los nodos colaboran en la obtención de los caminos mínimos.

- Inicialmente los nodos sólo conocen los costes de los enlaces a los nodos vecinos.
- En una serie de iteraciones los nodos comunican a sus vecinos toda la información de distancias que han recopilado.
- Con esa información los nodos computan distancias a nuevos nodos o actualizan distancias a nodos ya conocidos con valores menores.

Los mensajes de información que el nodo  $z$  comunica a sus vecinos son las distancias de los caminos que parten de dicho nodo,  $d_{z,i}$  con  $i = 1, \dots, n$  (siempre que le sean conocidas).

Supongamos que estamos en el nodo  $x$  con un enlace a un nodo vecino  $z$  de coste  $c_{x,z}$  y que el nodo  $z$  nos comunica la distancia  $d_{z,y}$  al nodo  $y$ . Nosotros podremos calcular la distancia  $D_{x,y}(z)$  de  $x$  a  $y$  saliendo por el vecino  $z$  usando la fórmula:

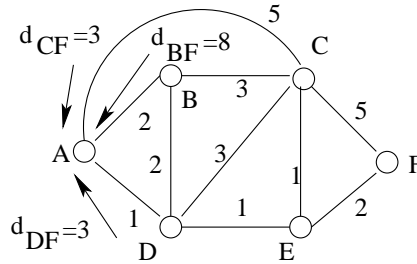
$$D_{x,y}(z) = c_{x,z} + d_{z,y}$$

distancia de  $x$  a  $y$  a través de la salida  $z$     coste del enlace que une  $x$  con  $z$     distancia de  $z$  a  $y$

Tal como hemos indicado en la fórmula, guardamos la identidad del vecino que nos ha proporcionado la información para que después la podamos poner en la tabla de rutas como línea de salida (router siguiente). Las cantidades  $D_{x,y}(z)$  se almacenan en una *tabla de distancias*, que toma la siguiente forma:

		Tabla de distancias del nodo x	
		distancia por	
		z	z'
destino	y	$D_{x,y}(z)$	$D_{x,y}(z')$
	y'	$D_{x,y'}(z)$	$D_{x,y'}(z')$

**Ejemplo.** Sea la red de la siguiente figura, donde los nodos B, C y D en una cierta etapa comunican el nodo A las distancias que conocen hacia el nodo F.



Entonces el nodo A puede calcular las distancias:

$$\begin{aligned}
 D_{A,F}(B) &= c_{A,B} + d_{B,F} = 2 + 8 = 10 \\
 D_{A,F}(C) &= c_{A,C} + d_{C,F} = 5 + 3 = 8 \\
 D_{A,F}(D) &= c_{A,D} + d_{D,F} = 1 + 3 = 4
 \end{aligned}$$

La tabla de distancias para el nodo A, en esta iteración queda:

		distancia por		
		B	C	D
destino	...	...	...	...
	F	10	8	4

El nodo A, a su vez, podría comunicar a sus vecinos en la siguiente iteración la distancia más corta que conoce hacia el nodo F, que es:  $d_{A,F} = \min_z D_{A,F}(z) = 4$ .

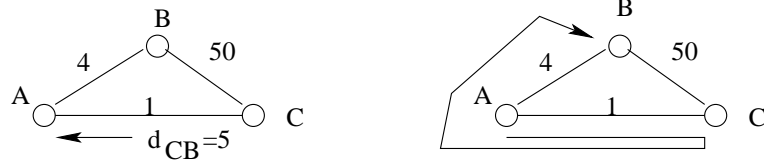
Después de una serie de iteraciones, no se produce ninguna actualización nueva en las tablas de distancias, por lo que se dice que el algoritmo ha convergido al resultado. En el caso de la red de la figura y para el nodo A, la tabla de distancias final es:

		distancia por		
		B	C	D
destino	B	2	8	3
	C	5	5	3
	D	4	7	1
	E	5	6	2
	F	7	8	4

A partir de la la tabla de distancias se puede obtener sin problemas la tabla de rutas, que en este ejemplo para el nodo A queda:

destino	salida
B	B
D	D
C	D
E	D
F	D

- Durante la ejecución del algoritmo puede suceder cosas curiosas, por ejemplo, sea la siguiente red:



$$D_{A,B}(C) = c_{A,C} + d_{C,B} = 1 + 5 = 6$$

(Obviamente esta ruta no entrará en las tablas de rutas, pero hay que tenerla en cuenta a lo hora de ejecutar el algoritmo).

### Características y comparación

- *Cambio del coste de un enlace.* Como es un algoritmo iterativo, la propagación de un cambio en el coste de un enlace se hace de nodo a nodo en varias iteraciones.

(1) Cuando se disminuye el coste de un enlace, los nodos se dan cuenta en seguida de que hay camino más corto y actualizan rápidamente su tablas.

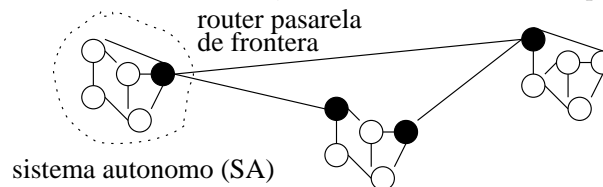
(2) Cuando se aumenta el coste de un enlace hay algunos problemas. Las tablas tardan mucho en actualizarse, porque caminos con distancias mayores a otros caminos ya existentes son despreciados por los nodos. Algunas técnicas pueden ayudar a que la convergencia del algoritmo sea más rápida. Consisten básicamente en mentir sobre el coste de los enlaces, por ejemplo, la técnica de la *inversa envenenada*.

- *Complejidad y velocidad de convergencia.* Al ser un algoritmo iterativo, el coste de cada iteración hay que multiplicarlo por el número de iteraciones. El problema que tiene el algoritmo VD es que puede tardar muchísimas iteraciones en converger. En este sentido es peor que el EE.
- *Robustez.* El algoritmo VD es menos robusto que el EE puesto que si un nodo calcula las distancias de forma incorrecta, todos los demás nodos usarán estos valores incorrectos.

### 4.2.3. Rutado jerárquico

- Las redes grandes como Internet se dividen en regiones denominadas *sistemas autónomos*. Un sistema autónomo suele estar operado por una empresa u organismo, por ejemplo el SA de la empresa telefónica en España o la red de investigación nacional, rediris.

Los routers conocen los detalles del encaminamiento en su región, pero desconocen los de las otras. El tráfico de la salida de la región se centraliza a través de uno o varios routers, denominados *routers pasarela de frontera*.



- Así se ejecutan dos algoritmos de rutado en los dos tipos de routers: *rutado intra-autónomo* dentro de la región y *rutado inter-autónomo* en los routers pasarela de frontera. Dentro de cada región el administrador del sistema puede elegir el protocolo de rutado que quiera, pero el rutado interautónomo ha de ser común para todos.

## 4.3. Encaminamiento en Internet

Vamos a ver los protocolos de rutado más usados en Internet. Como intra-autónomos, RIP y OSPF y como inter-autónomo, BGP.

Categoría	Protocolo	Tipo	Protocolos transporte/red
intra-autónomo	RIP	VD	UDP/IP (puerto 520)
	OSPF	EE	propio/IP (puerto 89)
inter-autónomo	BGP	VD	TCP/IP (puerto 179)

Como puede verse en la tabla, los protocolos de rutado son protocolos de la capa de aplicación (funcionan sobre sockets de forma normal).

#### 4.3.1. RIP (Protocolo de Información de Rutado)

- Es un rutado intra-SA de tipo VD (vector de distancias) en la cual todos los routers colaboran intercambiándose distancias. RIP considera que el costo de todos los enlaces es 1 y que la distancia máxima es 15, es decir, que RIP sólo puede encaminar paquetes a través de 15 routers intermedios.
- Los mensajes RIP se envían sólo a los vecinos. Cuando un router RIP se enciende o desea información sobre cierto destino se la solicita a sus vecinos con los *mensajes de petición RIP*.
- Los routers responden mediante los *mensajes de respuesta RIP*. Estos contienen una lista de hasta 25 redes destino dentro del SA con el formato:

red destino	métrica (distancia)
x	2
y	2
z	7

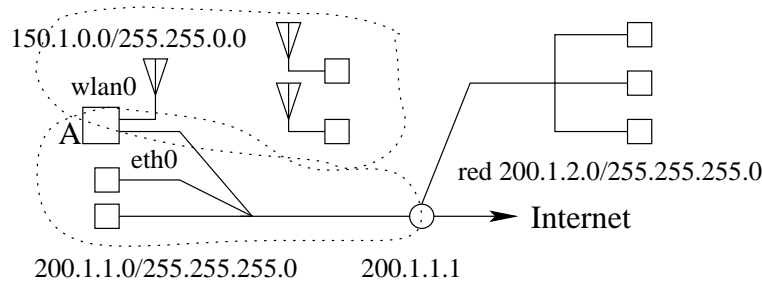
Los routers también envían automáticamente *mensajes de respuesta RIP* cada 30 segundos a sus vecinos. Si un router no tiene noticia de un vecino en los siguientes 180 segundos, considerará que está muerto, así que modifica sus tabla de rutas y anuncia esa información a los routers vecinos.

- En Linux, la tabla de rutas se obtiene con el comando *route*<sup>1</sup>, por ejemplo:

red destino		línea de salida		
IP	máscara	interface	gateway (prox.router)	métrica
150.1.0.0	255.255.0.0	wlan0	0.0.0.0 (*)	0
200.1.1.0	255.255.255.0	eth0	0.0.0.0 (*)	0
200.1.2.0	255.255.255.0	eth0	200.1.1.1	1
0.0.0.0 (default)	0.0.0.0	eth0	200.1.1.1	0

Normalmente, las entradas de la tabla se ponen al configurar las tarjetas de red, pero también podemos instalar el proceso de sistema *routed* para que nuestro ordenador aprenda las rutas a través de los mensajes RIP que envían los routers vecinos. En una ruta, si no se especifica la *gateway* es que la red está directamente conectada mientras que la ruta *default* se usa para destinos que no aparecen en las restantes líneas de la tabla.

<sup>1</sup>Este comando da la distancia (métrica) contando desde 0.



### 4.3.2. OSPF (Primero el Camino más Corto)

- OSPF (Shortest Path First). La O viene de Open e indica un algoritmo no patentado, es decir, libre (igual que la mayoría de los protocolos de Internet). Es un protocolo intra-SA más avanzado y pensado para suceder a RIP.
- Al contrario que RIP es un rutado EE (estado de enlaces). Los mensajes OSPF se difunden a todos los routers del SA y no sólo a los vecinos como en RIP. La información de los enlaces se difunde tanto cuando hay cambios como periódicamente aunque nada haya cambiado (al menos cada 30 minutos). De esta, forma cada router consigue la información completa de la red (del SA), y con ello puede ejecutar el algoritmo de Dijkstra.
- El protocolo OSPF comprueba que los enlaces sean operacionales enviando un mensaje HELLO a cada vecino. También existe la posibilidad de interrogar a un router vecino y obtener toda la base de datos que tenga este almacenada.
- El coste de los enlaces lo pone el administrador y no están limitados al valor 1 como en el caso de RIP.
- *Seguridad.* OSPF implementa un protocolo de transporte propio (no usa TCP ni UDP). Así todos los mensajes enviados van autenticados. Eso implica que sólo se va a tener en cuenta a los routers fiables. A los routers que no se han autenticado se los ignora.
- *Múltiples caminos de mismo coste.* Permite repartir el tráfico con el mismo destino a través de varios caminos alternativos. Las tablas de rutas tomarán la forma:

destino	salidas
A	B, C
B	B, D

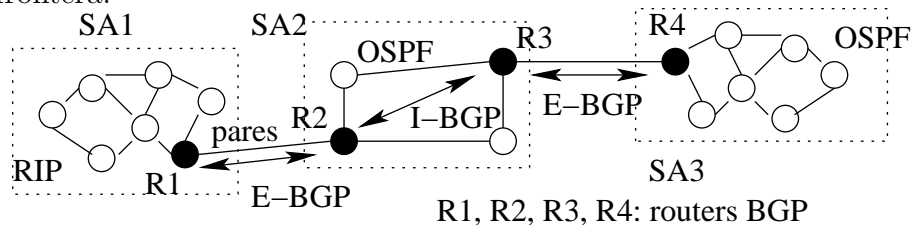
- *Soporte de jerarquía.* Debido al tamaño de Internet dos niveles de jerarquía no son suficientes, por lo que OSPF permite que cada SA se divida a su vez de áreas.

Cada área ejecuta el algoritmo de encaminamiento sobre los routers de ese área. Los mensajes OSPF no salen de cada área.

Las áreas se comunican entre sí a través de los *routers de frontera de área*. Cuando un paquete tiene un destino fuera del área se encamina primero hacia el router frontera de área. Los routers de frontera de área están interconectados entre sí, en lo que se denomina una *troncal* (backbone). Existe también un *router de frontera* (a secas) para sacar los paquetes del SA.

### 4.3.3. BGP (Protocolo de Pasarela de Frontera)

- BGP (Border Gateway Protocolo, Protocolo de pasarela de frontera) es el protocolo inter-SA estándar en Internet. Comunica routers pasarela de frontera.



- Cada SA está típicamente operado por una compañía, la cual usa en los routers el protocolo que quiere para transmitir dentro del SA, por ejemplo, OSPF o RIP. Uno o varios routers centralizan las transmisiones al exterior. Estos routers se denominan routers pasarela de frontera y usan BGP.
- Los routers BGP se comunican entre sí para intercambiar rutas. Hay comunicaciones de dos tipos:
  - entre routers BGP vecinos, denominados *pares* (o *iguales*) BGP.
  - entre routers pertenecientes al mismo SA (vecinos lógicos).

Dos routers BGP situados en el mismo SA aunque no estén conectados directamente por un enlace, se supone que tienen una conexión lógica entre ellos por lo que también se los considera vecinos.

Hay dos versiones del protocolo, el E-BGP (BGP externo) que se emplea para comunicar los pares BGP (pertenecientes a SA distintos) y el I-BGP (BGP interno) que comunica routers BGP dentro del mismo SA.

- Es un protocolo parecido al VD, aunque es mejor denominarlo *vector de rutas*. La razón es que en vez de limitarse a intercambiar información sobre distancias (como RIP), lo que hace es intercambiar rutas completas.

Ejemplo de mensaje en RIP (vector de distancias):

red destino	métrica
x	4



Ejemplo de mensaje en BGP (vector de rutas):

red destino	ruta
x	SA1/SA2/SA3/S4

La red destino se especifica en formato IP/máscara, por ejemplo 128.19.4.0/24.

- Cada SA se identifica por un *número de sistema autónomo*, *ASN*. Los números ASN, al igual que las direcciones IP, los facilita uno de los organismos de Internet. Típicamente cada proveedor de Internet tiene su propio número (o varios). Por ejemplo, la red nacional de investigación *rediris* tiene el número AS766, y telefónica el AS34036.
- Los administradores de los SA pueden seleccionar las políticas de rutado, por ejemplo, pueden decidir que no encaminarán ningún paquete (vetarán) a cierto proveedor de Internet. Esto lo pueden hacer simplemente haciendo que sus routers BGP no anuncien ninguna ruta a los routers BGP de la otra compañía.

## 4.4. El protocolo Internet (IP)

La capa de red en Internet consta de los siguientes elementos:

1. El *protocolo de red* denominado protocolo Internet (IP) que define el formato de las direcciones, los campos de los datagramas y las acciones que realizan los routers basándose en los campos de los datagramas.
2. El *Protocolo de rutado*.
3. El protocolo de mensajes de control de Internet (ICMP) usado para que los routers se comuniquen errores y otra información.

### 4.4.1. Direccionamiento IPv4

Las direcciones IPv4 son números de 4 bytes en formato de número y puntos. Todos los hosts y routers tienen una dirección IP por interface. Un interface es la unión del host o router con un enlace. Por ejemplo,

Interface	Denominación en Linux
tarjeta ethernet	eth0
cable serie	ppp0
cable USB	usb0
enlace wireless	wlan0

### Asignación de direcciones a una red

- Parte de los bits de una dirección IP identifican la red (son comunes para todos los hosts de la misma red) y los demás identifican el host dentro de la red. En vez de host sería más correcto decir interfaz, porque a los routers también se les da direcciones IP.
- Inicialmente las direcciones de red se clasificaron en clases. Se definen direcciones para tres clases de redes: A, B y C (que empiezan por 0, 10 y 110, respectivamente). Hay una cuarta clase, la D, que comienza por 1110 y es usada para multicast. Las direcciones que empiezan por 1111 están reservadas.

	red	host	
Clase A	0		$2^7=128$ redes, $2^{24}=16\text{M}$ hosts
	red	host	
Clase B	10		$2^{14}=16\text{k}$ redes, $2^{16}=64\text{k}$ hosts
	red	host	
Clase C	110		$2^{21}=2\text{M}$ redes, $2^8=256$ hosts

Para calcular el número de redes y de hosts de cada tipo, hemos tenido en cuenta que en informática  $1\text{k} = 2^{10} = 1024$  y  $1\text{M} = 2^{20} = 1048576$ . Los bytes 0 y 255 están reservados, por lo que deberíamos descontarlos.

La idea de las clases es que, a organizaciones grandes como a los gobiernos, se les daría una red de clase A, a organizaciones medianas, como grandes empresas, una de clase B, y a organizaciones pequeñas una red de clase C.

- El byte 0 se reserva para referenciar a la red (para rutado, con o sin máscara), y el 255 se usa para direcciones de difusión o broadcast (para enviar un datagrama a todos los hosts de la red). La dirección de broadcast se puede probar con `ping -b IP_broadcast`. Ejemplos:

Red	Difusión	Clase
193.144.84.0 (= 193.144.84)	193.144.84.255	C, 193=1100 0001
172.168.0.0 (= 172.168)	172.168.255.255	B, 172=1010 1100

- En 1993 se suprimieron las clases, estableciéndose la direcciones CIDR (Classless Inter-Domain Routing) y ahora se puede elegir cuántos bits forman las partes de red y hosts. Se indica mediante una máscara en la forma a.b.c.d/x (x bits de red y el resto de host). La clase A sería /8, la clase B /16 y la clase C /24, pero x puede ser cualquiera. Por ejemplo,

193.144.48.0/20				
193	144	48		0
193	144	0011	0000	0
red (20 bits)			host	

## Subredes

- Una red se puede dividir en subredes. Cada una de las subredes es equivalente a una red independiente.

Por ejemplo, para dividir una red en dos subredes aumentamos el campo de red de la dirección en 1 bit. Este bit puede tomar los valores 0 y 1, lo que origina las dos subredes que queremos:

193.144.48.0/20 en dos subredes				
193	144	0011-0-000	0	subred 193.144.48.0/21
193	144	0011-1-000	0	subred 193.144.56.0/21

Para hacer 8 subredes usaremos 3 bits:

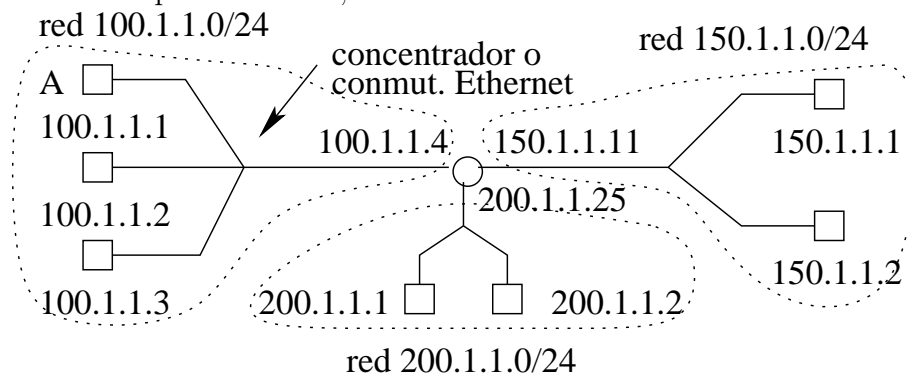
193.144.48.0/20 en 8 subredes				
193	144	0011-000-0	0	subred 193.144.48.0/23
193	144	0011-001-0	0	subred 193.144.50.0/23
...				
193	144	0011-111-0	0	subred 193.144.62.0/23

- Antiguamente las máscaras /x tenían un formato de 4 bytes compuestos de x 1s y el resto 0s, por ejemplo,

Formato actual	Formato antiguo
/16	255.255.0.0
/24	255.255.255.0
/28	255.255.255.224

## Ejemplo de red

Hay que dar una dirección IP a cada interface, es decir a cada extremo de un enlace en la capa de red, bien sea un host o un router. Como ejemplo, veamos 3 redes conectadas por un router,



Dos ejemplos de tablas de rutas, la del host A y la del router:

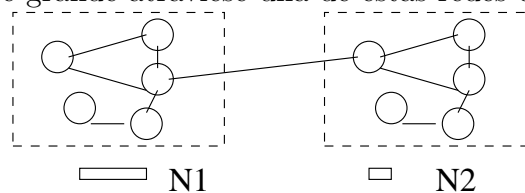
Tabla de rutas del host A			
destino	interface	gateway	métrica
100.1.1.0/24	eth0 (=100.1.1.1)	*	0
150.1.1.0/24	eth0 (=100.1.1.1)	100.1.1.4	1
200.1.1.0/24	eth0 (=100.1.1.1)	100.1.1.4	1

Tabla de rutas del router			
destino	interface	gateway	métrica
100.1.1.0/24	eth0 (=100.1.1.4)	*	0
150.1.1.0/24	eth1 (=150.1.1.11)	*	0
200.1.1.0/24	eth2 (=200.1.1.25)	*	0

#### 4.4.2. Formato del datagrama

1. **Fragmentación.** IP está pensado para que los datagramas atraviesen redes muy distintas. La capa de enlace de algunas redes sólo admite paquetes de tamaño muy pequeño, por ejemplo, Ethernet tiene un tamaño máximo de 1500 bytes y muchos enlaces de área extensa sólo admiten 576 bytes. Para que un paquete grande atravesase una de estas redes debe fragmentarse.



Los fragmentos no se vuelven a unir hasta llegar al destino. Es decir los router sólo fragmentan. Es responsabilidad de la capa de red del host destino recuperar todos los fragmentos, unirlos y pasarlos a la capa de transporte reconstruidos.





Como la fragmentación introduce complicaciones adicionales se intenta evitar la segmentación haciendo que TCP y UDP generen segmentos de pequeño tamaño (de 536 o 1460 bytes). Después se les añaden las cabeceras TCP e IP (en torno a 20+20 bytes) por lo que caben dentro de los 576 o 1500 bytes indicados antes. O sea, que típicamente cada segmento TCP o UDP genera un sólo datagrama.

Para hacer posible la fragmentación, en la cabecera IP hay tres campos: *identificación*, *indicadores* y *desplazamiento de fragmento*. El campo *indicadores* incluye tres bits. El primero no se utiliza, *NF* a 1 indica que el paquete no se fragmente, por lo que si es demasiado grande para atravesar una red debe desecharse. *MF* significa más fragmentos: todos los fragmentos, a excepción del último deben de tener este bit a 1.

2. **Numeración.** Hay dos niveles de numeración: uno para los datagramas (*identificador*) y otro para los fragmentos (*desplazamiento de fragmento*).

Por ejemplo: 0.0, 0.1, 0.2, ..., 1.0, 1.1, etc.

Dado que se proporcionan 13 bits, hay un máximo de  $2^{13} = 8192$  fragmentos por datagrama, dando así una longitud máxima de datagrama de 65 536 bytes, que coincide con el campo *longitud total*.

bits	4	4	8	16		
	version	lg. cab.	tipo servicio	longitud total		
	identificador			indicad.	despl. fragmento	
	tiempo vida		protocolo	suma comprobacion cabecera		
	direccion IP origen					
	direccion IP destino					
	opciones					
	datos					

3. **Versión.** Indica la versión IP. Actualmente es la versión 4 (IPv4), en el futuro se pasará a la 6 (IPv6).
4. **Longitud de la cabecera.** Puesto que no es constante, se mide en palabras de 32 bits. El valor mínimo es de 5 palabras (20 bytes).
5. **Longitud del datagrama.** Incluyendo cabecera y datos, se mide en bytes. El máximo es de 65 536 bytes, aunque, como hemos visto antes, lo normal es que sean más pequeños, 1500 bytes o frecuentemente 576 bytes.
6. **Tipo de servicio.** En la cabecera hay varios bits para indicar si se prefiere retardo mínimo, rutas de gran capacidad o de alta fiabilidad. Si el router tiene caminos alternativos para un mismo destino intentará elegir el más apropiado. Pero como sabemos, la capa de red no garantiza nada.
7. **Tiempo de vida (TTL, Time To Live).** El emisor lo inicia a un valor (el máximo es 255) y cada vez que es procesado por un router se decrementa en una unidad y cuando llega a 0 el datagrama se destruye.
8. **Protocolo.** Indica qué protocolo de transporte usa el paquete. Los números de protocolo se pueden consultar en `/etc/protocols`. TCP (6) y UDP (17) son algunos, pero existen más. Sólo se usan en origen y destino y constituyen el interface entre la capa de transporte y la de red. En el destino, la capa de red identifica a qué protocolo de la capa transporte deberá pasarse el datagrama.
9. **Suma de comprobación de la cabecera.** Permite comprobar errores, se recalcula en el receptor y si coincide con el enviado es que no ha habido errores. Si ha habido errores, habitualmente el datagrama se desecha. Es una suma considerando palabras de 16 bits y en complemento a 1. Este

campo puede cambiar si el datagrama se fragmenta. Aquí se calcula sólo la cabecera IP, en TCP y UDP se calcula sobre la cabecera TCP/UDP más los datos.

10. **Direcciones IP.** Hay dos: origen y destino y se emplean para encaminar los paquetes.
11. **Opciones.** En la cabecera hay un espacio para incorporar nuevas opciones. Una de ellas es la opción de que se grabe en la cabecera la lista de nodos por los que va pasando el datagrama. Otra opción es incluir en la cabecera la lista de los nodos por los que queramos que el datagrama pase (encaminamiento origen). El tamaño de la cabecera, es pues, variable.

### 4.4.3. IPv6

Como las direcciones IPv4 se están agotando ( $2^{32}$  no han resultado suficientes) se ha aprovechado la situación para promover también el cambio de formato del datagrama. Los cambios más significativos en IPv6 son los siguientes:

1. **Direcciones IPv6.** Se ha ampliado el número de bits de las direcciones de 32 a 128 bits. Ahora hay tantas direcciones que corresponden a varios miles por metro cuadrado del planeta: se ha asegurado por exceso de que no se vayan a agotar. El formato de dirección IPv6 es de 8 grupos de 4 dígitos hexadecimales ( $8 \times 4 \times 4 = 128$  bits). Se usan las máscaras de la misma forma que en IPv4 para separar las partes de red y de host. Por ejemplo:

dirección:	3ffe:ffff:100:1:2:3:4:5/48
máscara:	ffff:ffff:ffff:0000:0000:0000:0000:0000
red:	3ffe:ffff:0100:0000:0000:0000:0000:0000

Grupos de 4 ceros :0000: (o varios grupos de 4 ceros) se abrevian como ::, por ejemplo, fe80::20e:35ff:fe3e:c6f0:e4ff:fe0d/64 y ::1.

Aparte de la direcciones *unicast* (un host) y *multicast* (un conjunto de hosts) se incluye un nuevo tipo denominado *anycast* (el router entregará el paquete a uno cualquiera de los hosts de una red, lo cual es muy adecuado cuando hay un grupo de servidores espejo). Además, se ha eliminado el *broadcast* (todos los hosts de una red) y en su lugar deberá usarse *multicast*.

Las direcciones IPv6 que comienzan por el valor ff son multicast y el resto son direcciones unicast. Las direcciones anycast se obtienen a partir de las direcciones unicast poniendo a 0 el campo de host.

2. **Cabecera simplificada.** Se han eliminado varios campos:

*Opciones.* Ahora todas las cabeceras tienen el mismo tamaño, 40 bytes (ya no es necesario el campo *longitud de la cabecera*). Sin embargo, no han

desaparecido completamente, puesto que el campo de *siguiente cabecera* puede indicar una cabecera de un protocolo de opciones.

*Fragmentación/resamblado.* Si el paquete es demasiado grande para atravesar alguna red se desecha y se devuelve al emisor un mensaje ICMP que dice “paquete demasiado grande”. El emisor deberá reenviar los datos en paquetes más pequeños.

*Numeración de los paquetes.* Los paquetes ya no se numeran.

*Suma de comprobación.* Al eliminar las sumas se ahorra mucho procesamiento en el router. La comprobación todavía queda en la capa de transporte y posiblemente en la de enlace (por ejemplo, en Ethernet).

3. **Etiquetado de flujo.** Es el único campo nuevo y todavía no está claro cómo se utilizará. En principio, servirá para etiquetar a los flujos de datos, audio o vídeo generados por un mismo usuario o aplicación (todos los paquetes de un flujo tendrían la misma etiqueta).

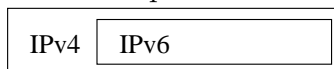
El nuevo formato es el siguiente:

	32 bits		
bytes	4 bits		8 bits
4	version	clase tráfico	etiqueta flujo (20 bits)
4	long. datos		sigui.cabec    limite saltos
16	direccion IPv6 origen (128 bits)		
16	direccion IPv6 destino		
total=40	datos		

Se mantienen los siguientes campos:

1. *Versión.* Ahora sería la 6.
2. *Clase de tráfico* (equivale al tipo de servicio en IPv4). Puede utilizarse para priorizar ciertas aplicaciones: rutas de bajo retardo, alta capacidad o fiables.
3. *Longitud de datos* (equivale a la longitud total en IPv4). Se mide en bytes sin contar la cabecera.
4. *Siguiente cabecera* (equivale al protocolo en IPv4). Identifica a qué protocolo pertenece la siguiente cabecera, que puede ser un campo de opciones o la capa de transporte a la que hay que entregar los datos, TCP, UDP, etc.
5. *Límite de saltos* (equivale al tiempo de vida en IPv4). Se decrementa en cada salto, cuando llega a cero, el paquete se elimina.

En cuanto a la transición de IPv4 a IPv6, ya se ha reconocido que el cambio tiene que ser gradual y que durante un tiempo van a coexistir las dos. La solución es la *tunelización*, que consiste en encapsular los paquetes IPv6 dentro de paquetes IPv4 normales, añadiéndoles una cabecera IPv4 redundante. La tunelización se empleará cuando los routers por los que tiene que pasar el paquete no admiten IPv6. Las cabeceras IPv4 se quitarán cuando se llegue a una zona en la que los routers puedan manejar de nuevo IPv6. Las direcciones origen y destino de IPv4 que se insertan son las de los routers que hacen la tunelización.



#### 4.4.4. ICMP, Protocolo Mensajes de Control de Internet

ICMP se emplea para que hosts y routers puedan informarse sobre errores o sobre el estado de la red.

Es un protocolo que funciona en la capa de transporte sobre el protocolo de red IP. Contienen muy poca cosa: dos campos para indicar el tipo y el código del mensaje y los 8 primeros bytes del datagrama que causó el envío del mensaje ICMP.

Tipo ICMP	Código	Descripción
0	0	respuesta de eco
3	0	red destino inaccesible
3	1	host destino inaccesible
3	2	protocolo destino inaccesible
3	3	puerto destino inaccesible
3	6	red destino desconocida
3	7	host destino desconocido
4	0	apaciguar fuente
8	0	petición de eco
9	0	anuncio de router
10	0	descubrimiento de router
11	0	TTL espirado
12	0	mala cabecera

Algunos ejemplos:

- El ejemplo típico es el mensaje de error *Destination host unreachable*. Este mensaje ICMP lo origina un router que fue incapaz de encontrar el camino al destino y lo devuelve al host origen. El host origen pasa su código de error a la capa de transporte y la capa de transporte se lo pasa a la aplicación, que es la que nos lo muestra en pantalla.
- Otro ejemplo típico es el comando *ping* que se utiliza para obtener el tiempo de ida y vuelta a un determinado destino. El mensaje ICMP de solicitud se denomina *solicitud de eco* y el de respuesta *respuesta de eco*.



- Otro ejemplo es *TTL espirado*. Cuando en un paquete el tiempo de vida (TTL) llega a 0 sin llegar al destino, el router lo elimina y devuelve este mensaje ICMP al origen.

#### 4.4.5. DHCP

Una vez que se nos ha asignado un bloque de direcciones IP para utilizar en nuestra red, tenemos dos opciones para distribuir las entre los hosts:

- *Estáticamente*. Indicando en el momento de instalación del sistema operativo la dirección IP que corresponde a cada host.
- *Dinámicamente*. Usando el *Protocolo de configuración dinámica de hosts* (DHCP). Cada vez que un host arranca le solicita al servidor que le proporcione una IP temporal. Esta IP será válida durante un cierto tiempo o mientras el host esté encendido. Si el host se reinicia tendrá que pedir una nueva IP, que seguramente será distinta a la que tenía antes. Este protocolo se suele usar cuando los ISP no tienen direcciones IP para todos sus abonados que se conectan por modem o ADSL. También se suele usar en las conexiones inalámbricas.

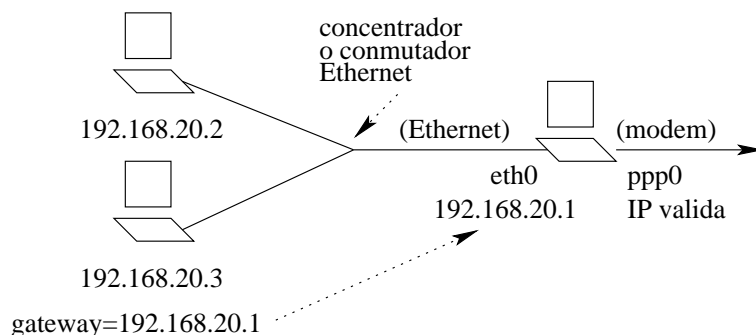
El protocolo DHCP comprende 4 pasos:

1. *Descubrimiento de un servidor DHCP*. Lo primero que tiene que hacer un host para obtener una dirección por DHCP es encontrar el servidor que va proporciona las direcciones. Este es el mensaje de descubrimiento. ¿Qué direcciones IP origen y destino tiene este mensaje? Como dirección destino usa la de difusión 255.255.255.255 que es la que atienden todos los hosts y como dirección origen 0.0.0.0.
2. *Ofrecimiento del servicio DHCP*. El servidor responde con un mensaje que incluye una dirección IP y su máscara de red junto con un tiempo de concesión de la IP, que irá desde varias horas a varios días.
3. *Petición DHCP*. Si hay varias ofertas, el cliente solicita una.
4. *ACK DHCP*. El servidor responde reconociendo la petición.

#### 4.4.6. NAT, Traducción de direcciones de red

¿Qué podemos hacer si nuestro proveedor de Internet nos da una única IP pero nosotros tenemos varios ordenadores en casa? Obviamente no podemos usar a la vez la misma IP en varios ordenadores. Pero podemos usar una solución que se denomina NAT, *traducción de direcciones de red*.

El esquema es de la siguiente forma:



- Uno de los ordenadores, el servidor NAT, funcionará como el router que nos conectará la red privada con Internet. Puede usarse cualquier tipo de conexión con el exterior: modem, ADSL, RDSI, PPP, red, etc. A este ordenador le asignaremos la dirección IP válida.

Es muy fácil utilizar un ordenador con Linux para que funcione como router y servidor NAT. Básicamente consiste en configurar y arrancar un programa, por ejemplo, *iptables* o *ipchains*.

- Al resto de los ordenadores les asignaremos direcciones IP privadas. Mundialmente se han reservado un conjunto de direcciones IP válidas sólo para redes internas. Estas son:

Clase A	10.0.0.0/8
Clase B (varias)	172.16.0.0/12
Clase C (varias)	192.168.0.0/16

No hay ningún peligro en usar estas direcciones, pues aunque accidentalmente salgan de nuestra red, los routers de Internet ignorarán los paquetes con estas direcciones.

Estos ordenadores estarán conectados al servidor a través de una red, Ethernet, por ejemplo.

- Como podemos ver, el servidor NAT necesita dos interfaces y dos direcciones IP, uno de ellos (por ejemplo *ppp0*) con la dirección IP válida para conectar al exterior, mientras que el otro (por ejemplo *eth0*) con una dirección privada para conectar con la red interna.

El resto de los ordenadores sólo se conecta mediante el interface de la red interna y cada uno de ellos usa una dirección privada distinta. Cuando configuremos estos ordenadores, como gateway tendremos que indicarles la dirección IP privada del servidor NAT.

- ¿Qué hace el servidor NAT?, pues actúa como router pasando las transmisiones de los ordenadores al exterior, pero también como traductor, cambiando las direcciones privadas por la IP válida.

También toma nota de la conexión TCP o UDP que se abre (el puerto origen), para que cuando llegue la respuesta, pueda devolvérsela al ordenador adecuado. Como podría haber repeticiones en los puertos origen, el servidor los cambia para que sean únicos. La tabla tiene por un lado la dirección IP interna y el puerto original, y por el otro el puerto modificado.

IP origen		puerto destino		puerto origen	
192.168.20.2	80	8000			
192.168.20.3	80	8001			
192.168.20.4	80	8000			

NAT →

IP valida	80	8000			
IP valida	80	8001			
IP valida	80	8002			

Cuando llega una respuesta, el servidor le examina el puerto y obtiene de la tabla la IP privada y el puerto original y se entrega al ordenador que hizo la petición.

puerto del paquete	host original	puerto original
...		