

Capa de RED

La capa de red esta formada por

- Subredes: formada por enrutadores interconectados
- Hosts o LANs conectadas a subred

Varias subredes se conectan a travez de puertras de enlace

Existen varias alternativas para mandar conjuntos de paquetes desde un host a otro

- Ruta fija (servicio orientado a conexion)
- Ruta variable (servicio no orientado a conexion)

Servicio no orientado a la conexión:

- Alentado por la comunidad de internet
- Los paquetes se enrutan de manera independiente.
 - La ruta a usar entre los hosts va a **cambiar cada cierto tiempo**.
 - Cada paquete debe llevar una dirección de destino completa.
- Nomenclatura usada:
 - Paquetes = **datagramas**
 - Subredes = **subredes de datagramas**

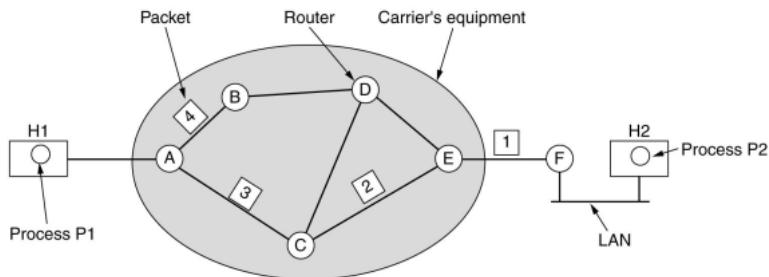
Diseño de la tabla de un enrutador.

Suponemos:

- Existe un procedimiento que dada la dirección del host de destino me retorna dirección del enrutador de destino (al cual está conectado host de destino).
 - Despues veremos como se puede hacer esto.
- Enrutador de destino **sabe cómo entregar** paquete a host de destino (por más que el host de destino esté en una LAN).

Tabla del enrutador

- La tabla de enrutamiento solo necesita entradas para los enrutadores de la subred.
- **entrada de tabla** de enrutador formada por filas:
 - <enrutador de destino, línea de salida>
 - La línea de salida es la dirección de un enrutador.



A's table

	initially	later
A	-	-
B	B	B
C	C	C
D	B	B
E	C	B
F	C	B

C's table

A	A
B	A
C	-
D	D
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	-
F	F

Dest. Line

Cuando llega un paquete a un enrutador:

1. Se lo almacena y se comprueba que llegó bien
2. Se determina el enrutador de destino asociado al host de destino
3. Se usa fila de ese enrutador de destino para reenviar el paquete por línea de salida de esa fila.

Cada enrutador, tiene asociada una salida a un enrutador destino. En este ejemplo si H1 quiere mandar un mensaje a H2, primero debería buscar la forma de mandar el paquete al enrutador F

De esa forma A se fija en la entrada de F en su tabla de enrutador.

Podemos pensar que **dirección de un host** es un número con dos partes:

- <dirección de red, número de máquina>
- La **dirección de red** sirve para identificar una red (p.ej. una red local).
- El **número de máquina** sirve para identificar una máquina dentro de la red.
- P.ej: direcciones de 8 bits, red de 4 máquinas viene dada por las direcciones: 11010000, 11010001, 11010010 y 11010011. (dirección de red es 110100)
- IP respeta esta convención pero con direcciones de 32 bits.

Los enrutadores que están conectados a un host, también forman parte de esa red.

Es decir, tienen el mismo valor en la parte de dirección de red.

Todo host destino va a tener un enrutador con misma dirección red.

Ese es el enrutador destino a usar para llegar al host destino

Dado un host destino para mandar un mensaje, para encontrar el enrutador destino apropiado. Hay que buscar el enrutador destino cuya dirección concuerde con la mayor cantidad de bits desde la izquierda con la dirección de host destino.

Servicio orientado a la conexión

- Alentado por las **compañías telefónicas** (p.ej. ATM)
- Todos los paquetes se mandan por la misma ruta.
- Trabajo a realizar antes de mandar paquetes:**
 - Hay que **configurar** una ruta del host de origen al de destino.
 - Esto se llama crear una **conexión**.
 - **Círculo virtual (CV)** = conexión
- Cada paquete lleva un **identificador** que indica a cual CV pertenece.
- Cuando no se necesita enviar más paquetes se **libera la conexión**. Al hacer eso también se termina el CV.

Se elige una ruta de la maquina de origen a la de destino y se almacena en tablas dentro de los enrutadores.

Cuando crece mucho el tamaño de las subredes

- También lo hacen las tablas de enrutamiento

Consecuencias de tener tablas de enrutamiento grandes:

- Estas tablas consumen memoria del enrutador, necesitan más tiempo de CPU para examinarlas.

Que hacer cuando las tablas crecen mucho?

Solución: enrutamiento jerárquico

- Los enrutadores se dividen en **regiones**.
- Un enrutador sabe cómo enrutar paquetes a destinos en su región.
- También sabe cómo enrutar a otras regiones.
- Pero no sabe nada de la estructura interna de las regiones en las que no está.

Precio a pagar con enrutamiento jerárquico: una *longitud de ruta mayor* (no se puede aspirar a encontrar la mejor ruta).

Las tablas de enrutamiento jerárquico tendrán una entrada para cada enrutador local. Y luego una entrada para las demás regiones en donde no está el enrutador.

Cuando las redes son muy grandes se agrupan las regiones en clusters, los clusters en zonas y las zonas en grupos.

Las tablas de enrutamiento son creadas por algoritmos de enrutamiento

Algoritmos de cálculo de la ruta más corta entre dos nodos.

- Uno de ellos es el algoritmo de **Dijkstra** (1959).
 - Dado grafo conexo con costos en los enlaces, y nodo n en el grafo, obtiene **árbol de caminos más cortos** desde n hacia todos los demás nodos.
 - El árbol de caminos más cortos se representa con un **mapeo** donde para cada nodo del grafo de la subred asigna su parente (en el árbol de caminos más cortos).
 - Repasar los detalles del algoritmo de Dijkstra visto en algoritmos 2.

Inundación

Idea de inundación: para enviar un paquete de un origen u a un destino v los caminos usados son aquellos que respetan las siguientes reglas:

- u manda el mensaje por todas las líneas de salida.
- Cada paquete que llega a un enrutador distinto de v se reenvía por cada una de las líneas excepto aquella por la que llegó.

Genera muchos paquetes duplicados

Por ello se limita la inundacion

Cada enrutador recuerda que paquetes ya envio, de modo que si le vuelve a llegar no lo acepta

Refinamiento de la solución anterior:

- El enrutador de origen pone un **número de secuencia** en cada paquete que recibe de sus hosts (así se distingue entre paquetes distintos del mismo enrutador de origen).
- Un enrutador recuerda para cada enrutador de origen los números de secuencia recibidos – i.e. pares <enrutador de origen, n° secuencia>
- Si llega un paquete a un enrutador con par <enrutador de origen, número de secuencia> recibido antes, no se lo reenvía.
- Agregar una columna **contador** que indica el mayor número de secuencia tal que:
- Illegaron paquetes con todos los números de secuencia anteriores desde ese enrutador de origen.

Ademas cada paquete, tiene un contador de saltos, tal que en cada enrutador se le resta 1, y cuando llega a 0 se lo descarta.

Nº enrutador de origen	Contador	Lista de N° de secuencia vistos
		→ □ □ → □ □ → □ □
		→ □ □ → □ □

Enrutamiento de estado de enlace

Una idea que implementa este algoritmo, es aplicar Dijkstra periodicamente actualizando las tablas de enrutamiento segun los cambios en la topología de la red.

Enrutamiento de estado de enlace (Link state routing -LSR)

- En cada enrutador usar **algoritmo de Dijkstra** para encontrar la ruta más corta de un enrutador a los demás enrutadores.
- La topología y retardos en las líneas se distribuyen a cada enrutador.
- Este algoritmo es valioso porque:
 - Responde rápido frente a cambios en la topología de la red.
 - Es ampliamente usado en Internet (como parte del protocolo OSPF)

¿Qué tareas debe hacer un enrutador LSR?

1. Descubrir sus vecinos
2. Medir el costo a cada uno de sus vecinos
3. Construir un paquete diciendo lo que ha aprendido
4. Enviar este paquete a todos los demás enrutadores
5. Computar el camino más corto a cada uno de los otros enrutadores

Un enrutador envia paquetes Hello a cada linea de salida y espera una respuesta donde un enrutador del otro extremo le indica quien es.

Una vez que un enrutador conoce todos sus vecinos. Envía paquetes ECHO y el otro extremo debe reenviarlos inmediatamente. Por medio de temporizadores se calcula ida y vuelta y se /2

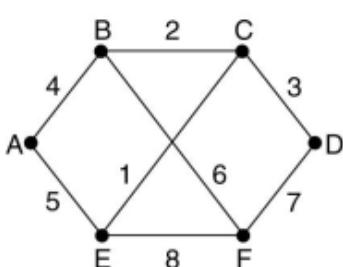
Cada enrutador construye un **paquete de estado de enlace (LSP)**

¿Qué datos poner en el LSP?

- Identidad del emisor
- Número de secuencia
- Edad
- Lista de <vecino, retardo al vecino>

¿Cuándo se pueden construir los LSP?

- Construirlos a intervalos regulares.
- Construirlos cuando ocurra un evento significativo, como la caída o la reactivación de la línea o de un vecino, o el cambio apreciable de sus propiedades.



(a)

Link	State	Packets
A	B	E
Seq.	Seq.	Seq.
Age	Age	Age
B 4	A 4	A 5
E 5	C 2	C 1
	D 3	F 7
	F 6	F 8
	E 1	E 8

(b)

Subred

LSP para cada nodo

Se utiliza inundacion para distribuir los LSP.

Todos los enrutadores se van mandando sus paquetes LSP, para asi todos conocer la red. Y cada enrutador lleva un registro de los paquetes difundidos. Ya que como vimos, cada LSP contiene un numero de secuencia que se aumenta cada vez que se envia desde su enrutador de origen.

Cada enrutador lleva registro de los pares <enrutador de origen,secuencia>

Se utiliza el mismo criterio que en inundacion. Si:

- seq menor, se descarta por obsoleto.
- seq igual, se descarta por duplicado.
- seq mayor, se guarda la info necesaria y se reenvia por todas las lineas.

Se puede construir la tabla de enrutamiento de un enrutador

- una vez que el enrutador ha acumulado un grupo completo de paquetes de estado del enlace

Primero usando los LSP construir el grafo de la subred completa.

- Cada enlace se representa dos veces, una para cada dirección.
- Los dos valores pueden promediarse o usarse por separado.

Se ejecuta el **algoritmo de Dijkstra** para construir la ruta más corta a todos los destinos posibles.

Con los resultados del mismo se actualiza la tabla de enrutamiento.

Como se afrontan los posibles problemas:

-para evitar reinicio de num de seq, se usan 32 bits

-para evitar seq corruptos (bits cambiados) se confirma recepcion de todos los paquetes de estado de enlace (LSP)

-si se cae un enrutador de origen, su seq volvera a 0. Cuando se actualicen las tablas de enrutamiento y se deberan mandar paquetes Hello, este no respondera entonces esta caido. esa info se propaga y luego toda la info asociada a ese enrutador expira.

-Ningun paquete perdido sobrevive un tiempo indefinido pq tienen un paquete edad, que se reduce por cada segundo y cuando llega a 0, todo enrutador lo descartara.

Ahora vemos cómo hacer el algoritmo de inundación de paquetes de estado de enlace más eficiente:

- Una vez que un paquete de estado del enlace llega a un enrutador para ser inundado, no se encola para transmisión inmediata. En vez de ello, entra en un **búfer de almacenamiento** donde espera un tiempo breve.
- Si antes de transmitirlo, llega otro paquete de estado del enlace proveniente del mismo origen, se comparan sus números de secuencia.
 - Si son iguales, se descarta el duplicado.
 - Si son diferentes, se desecha el más viejo.

Recordamos que la idea de usar LSP, es. Por inundacion todos los enrutadores conocen los LSP, cada uno arma la red. y ejecutan dijkstra para conocer los mejores caminos. luego no se usa inundacion para enviar paquetes, sino esos caminos que estan en las tablas de enrutamiento generadas.

El **buffer de paquetes para un enrutador** contiene una celda por cada paquete de estado de enlace recién llegado, pero aun no procesado por completo.

Una fila de la tabla del búfer de paquetes de un enrutador contiene:

- Origen del paquete, número de secuencia, edad, datos de los estados de enlaces.
- Banderas** que pueden ser:
 - **Banderas de confirmación de recepción**: indica a dónde tiene que enviarse la confirmación de recepción del paquete.
 - **Banderas de envío**: significan que el paquete debe enviarse a través de las líneas indicadas.
 - Si llega un duplicado mientras el original aún está en el búfer, los bits de las banderas tienen que cambiar.

Source	Seq.	Age	Send flags			ACK flags			Data
			A	C	F	A	C	F	
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Enrutamiento de vector de distancia

Cada enrutador mantiene una **tabla de enrutamiento (o de reenvío)** indexada por cada enrutador en la subred.

- Cada entrada comprende: la **línea preferida de salida hacia ese destino y una estimación del tiempo o distancia a ese destino**.

A partir de su tabla de enrutamiento un enrutador E puede obtener **un vector de distancia** que contiene una lista de pares <destino, retardo estimado>

El retardo de un enrutador a un vecino suyo, puede medirlo con

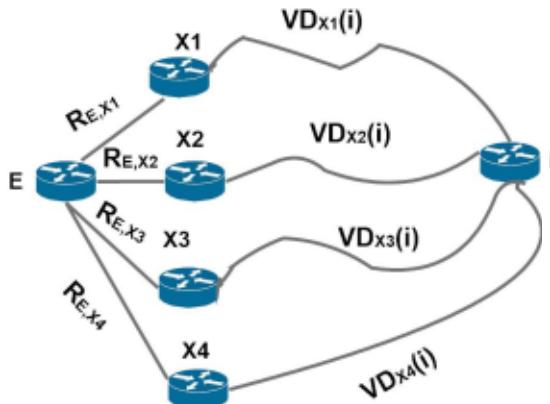
- **paquetes de ECO** que el receptor simplemente marca con la hora y los regresa tan rápido como puede.

Un poco de notación:

- El vector de distancia del enrutador X se denota con VD_X .
- VD_X es una función: $VD_X(i)$ es la 'distancia estimada' para llegar al enrutador i desde X .
- Si X vecino de E , el retardo de E a X se denota con $R_{E,X}$.
 - se usa paquete ECO para obtenerlo.

Entonces la distancia estimada desde E enrutador a i a través de X es:

$$R_{E,X} + VD_X(i).$$



En este caso el enrutador E estima la distancia al enrutador i de la forma:
 $d(E,i) = \min\{R_{E,X} + VD_X(i) \mid X \text{ vecino de } E\}$

el Mejor vecino para ir de E a i se def como
 $MV(E,i) = \text{elegir}\{V: R_{E,X} + VD_X(i) = d(E,i)\}$

¿Cómo se actualiza **tabla de enrutamiento** de E ?

- E recibió de todo vecino X suyo: VD_X y $R_{E,X}$
- La tabla de enrutamiento de E en la fila del enrutador de destino i va a tener los valores: $d(E,i)$ y $MV(E,i)$.
- Observar que la vieja tabla de enrutamiento no se usa en este cálculo.

Tengo estimación $R_{E,X_n} + VD_{X_n}(i)$ de camino más corto de E a i que pasa por X_n , para todo n en $\{1, \dots, 4\}$.

¿cuál es la mejor de esas estimaciones?

Aquella que tiene el menor valor en

$$\{R_{E,X_1} + VD_{X_1}(i), R_{E,X_2} + VD_{X_2}(i), R_{E,X_3} + VD_{X_3}(i), R_{E,X_4} + VD_{X_4}(i)\}$$

El vecino de E con la mejor de esas estimaciones conviene que sea la línea de salida a usar desde E para ir a i .

Considere un enrutador cuya mejor ruta al destino X es larga. Si en el siguiente intercambio el vecino A informa repentinamente un retardo corto a X ,

- el enrutador simplemente se conmuta a modo de usar la línea a A para enviar tráfico hasta X .

Supongamos que la métrica de retardo es el número de saltos.

- Las buenas noticias se difunden a razón de un salto por intercambio.
- En una subred cuya ruta mayor tiene una longitud de N saltos, en un lapso de N intercambios todo el mundo sabrá sobre las líneas y enrutadores recientemente revividos.

La razón de porqué las malas noticias viajan con lentitud es: ningún enrutador jamás tiene un valor mayor en más de una unidad que el mínimo de todos sus vecinos.

- Gradualmente todos los enrutadores elevan cuentas hacia el infinito, pero el número de intercambios requeridos depende del valor numérico usado para el **infinito**.
 - Si la métrica usada es el número de saltos, es prudente hacer que el infinito sea igual a la ruta más larga más 1.

Si la métrica es el retardo de tiempo no hay un límite superior bien definido,

- se necesita un valor alto para evitar que una ruta con un retardo grande sea tratada como si estuviera desactivada.

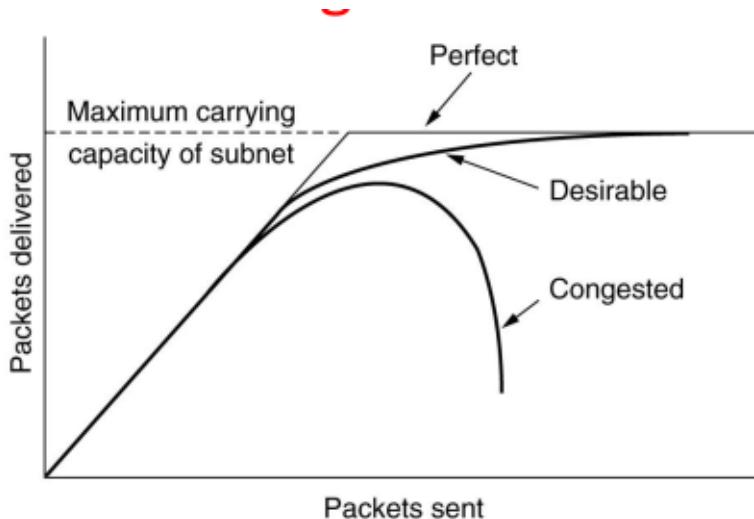
Este es el **problema de la cuenta hasta el infinito**.

- Se han hecho varios intentos para resolverlo, pero ninguno funciona bien en general.
- La esencia del problema consiste en que cuando X indica $VD_X(i)$ a E , E no tiene forma de saber si el destino i está en alguna ruta en funcionamiento.

Control de congestión

Los enrutadores tienen un buffer antes de cada enlace, y estos son de tamaño finito, por lo que si llega un paquete cuando el bufer esta lleno, este se pierde. Y el paquete debe ser retransmitido por el enrutador previo o el host emisor.

Hay un buffer por cada linea de salida, es decir que si llegan muchos paquetes que desean ir por la misma linea de salida, entonces se acomulan en una cola y varios se perderan.



Cuando hay demasiado tráfico, surge la congestión y el desempeño se degrada rápidamente

La meta del control de congestión es

- asegurar que la subred sea capaz de transportar el tráfico ofrecido

Ya vimos que en la capa de transporte se enteran tarde de la congestión (expiracion de temporizador o 3ack duplicados) y no tienen control sobre que paquetes se pierden. Por ello el control de congestión en capa de red, ayud a los host a controlarlo mejor.

La principal solucion a controlar la congestión es disminuir la carga en la subred. Como?

Regulación del tráfico

- hacer que hosts responsables de la congestión se enteren más rápido (que con protocolos de TCP) de la congestión y reduzcan su tasa de transferencia

Desprendimiento de carga

- Enrutadores descartan paquetes inteligentemente antes que se saturen búferes.

Problema: ¿Cómo puede hacer un enrutador para darse cuenta si tiene algún puerto de salida congestionado?

Solución: Cada enrutador monitorea la **demora de la cola** de línea de salida.

- Asociar a cada línea: $d = \text{demora reciente de cola de esta línea}.$
- Tomar periódicamente una muestra de la **longitud de cola instantánea de la línea**, s
- Actualizar d periódicamente usando:

$$d_{nvo} = a d_{ant} + (1-a) * s$$

donde a determina la rapidez con que el enrutador olvida la historia reciente.

Siempre que d rebasa un **umbral**, la línea de salida entra un **estado de advertencia**.

- Cada paquete nuevo que llega se revisa para ver si su línea de salida está en estado de advertencia.
- Si es así, se realiza alguna acción.

Como la congestión es algo que solo ocurre en los enrutadores, ellos son los encargados de avisarle a un host que hay congestión.

Método de paquetes reguladores.

1. Usar **paquetes reguladores** si línea de salida en estado de advertencia.
 - o El enrutador regresa un **paquete regulador (PR)** al host de origen, proporcionándole el destino encontrado en el paquete.
2. **Para que el paquete original no genere más PR más adelante en la ruta**
 - o en el paquete original se activa un bit del encabezado y después se reenvía.
3. **El PR** le pide al host de origen que reduzca en un porcentaje X el tráfico enviado al destino especificado.
4. El host ignora los PR que se refieran a ese destino por un intervalo fijo.
5. Una vez que haya expirado ese tiempo, el host escucha más PR durante un intervalo I .
 - Si llega alguno el host reduce el flujo aun más y comienza a ignorar nuevamente los PR.
 - **Si no llega ningún PR durante I** el host incrementa el flujo.

Problema: Cuando hay altas velocidades o distancias grandes, este metodo tiene una reaccion muy lenta.

Solución: Método de Paquetes reguladores de salto por salto. Hacer que el paquete regulador ejerza su efecto en cada salto que da.

- Cuando el paquete regulador llega a un enrutador F, se le obliga a F a reducir el flujo al siguiente enrutador D (F deberá destinar más búferes al flujo).
- Luego el paquete regulador llega al enrutador E anterior a F e indica a E que reduzca el flujo a F. Esto impone una mayor carga a los búferes de E, pero da un alivio inmediato a F. Y se sigue así sucesivamente.

Método de bit de advertencia. Señalar el estado de advertencia activando un bit especial en el encabezado del paquete.

- Cuando el paquete llega a su destino, la entidad transportadora copia el bit en la siguiente confirmación de recepción que se regresa al origen.
- A continuación el origen reduce el tráfico.
- Mientras el enrutador está en estado de advertencia, continua activando el bit de advertencia, lo que significa que el origen continua obteniendo confirmaciones de recepción con dicho bit activado.

El origen monitorea la fracción de confirmaciones de recepción con el bit activado y ajusta su tasa de transmisión de manera acorde.

- En tanto los bits de advertencia continúan fluyendo, el origen continua disminuyendo su tasa de transmisión.

Cuando la tasa de transmisión disminuye lo suficiente, el origen incrementa su tasa de transmisión.

- Debido a que cada enrutador a lo largo de la ruta puede activar el bit de advertencia, el tráfico se incrementa solo cuando no había enrutadores con problemas.

Una implementación de bit de advertencia usada por TCP es **ECN (Explicit Congestion Notification)**:

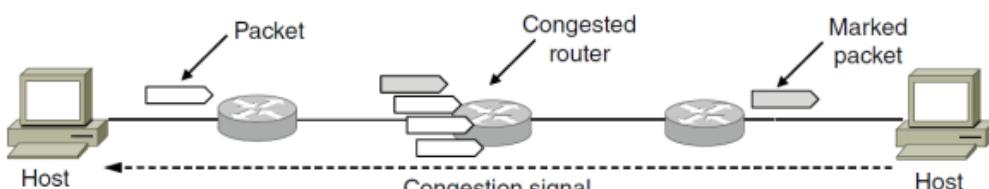
- Se usa en TCP/IP.
- Se marcan 2 bits en el encabezado IP con distintos fines:
 - 00: transporte no capaz de ECN
 - 10: transporte capaz de ECN, **ECT(0)**
 - 01: transporte capaz de ECN, **ECT(1)**
 - 11: congestión encontrada, **CE**
- Si ambos extremos soportan ECN mandan sus paquetes con ECT(0) y ECT(1) respectivamente.
- Si paquete atraviesa cola congestionada y el enrutador soporta ECN, se cambia código en el paquete a CE para avisar al receptor de la congestión.

• **Secuencia de ejecución de ECN típica:**

1. Se negocia ECN en conexión TCP
2. Emisor manda paquete IP *P* con ECT(0)
3. *P* llega a enrutador congestionado que soporta ECN y enrutador marca *P* con CE.
4. Receptor recibe *P* con CE y manda segmento *Q* (con ACK de *P*) de vuelta usando bandera ECE prendida.
5. Emisor recibe *Q* con ECE prendido, entonces emisor reduce ventana de congestión.
6. Emisor manda siguiente segmento al otro extremo usando bandera CWR prendida para confirmar recepción de aviso de congestión.

El uso de este protocolo es opcional, y se negocia al momento de establecer la conexión entre el emisor y el receptor.

Nota: Se continua transmitiendo segmentos con ECE prendido hasta recibir segmento con CWR prendido.



La otra alternativa a controlar la congestión es el desprendimiento de carga para evitar perder paquetes de manera indiscriminada.

La idea es eliminar paquetes cuando una cola está en estado de advertencia en una línea de salida.

Según la aplicación hay 2 estrategias:

-Vino: descartar los paquetes más nuevos, Ej: en transferencia de archivos

-Leche: descartar los más viejos, Ej: en multimedia.

También se puede crear un sistema de prioridades, donde cada paquete que entra se le asigna una prioridad, tal que cuando se entra en advertencia se eliminan los paquetes de menor prioridad.

La mejor estrategia es mezclar ambas estrategias. Desprendimiento de carga junto reducción de tráfico. De tal forma que la respuesta a paquetes perdidos por desprendimiento sea que el host origen disminuya la tasa de transferencia.

Implementación: Algoritmo de detección temprana aleatoria (RED).

- Para detectar cuándo comenzar a descartar paquetes, los enrutadores mantienen un **promedio móvil de sus longitudes de cola**.
- **Cuando este promedio de una cola C sobrepasa el umbral**
 - Una **pequeña fracción** de los paquetes son descartados al azar.
- **Con cada uno de esos paquetes:**
 1. El enrutador **elige un paquete al azar** de C .
 2. Se descarta el paquete seleccionado.
 3. El origen notará falta de ACK y la **capa de transporte** disminuirá la velocidad de transmisión.

Consecuencias de elegir paquetes al azar:

- hace más probable que los hosts emisores más rápidos pierdan un paquete, lo noten, y reduzcan su tasa de transferencia.

El método RED se usa en internet cuando los hosts no pueden recibir señales explícitas de congestión.

- Tanembaum dice que la mayoría de los hosts de internet no reciben mensajes explícitos de congestión de los enrutadores.

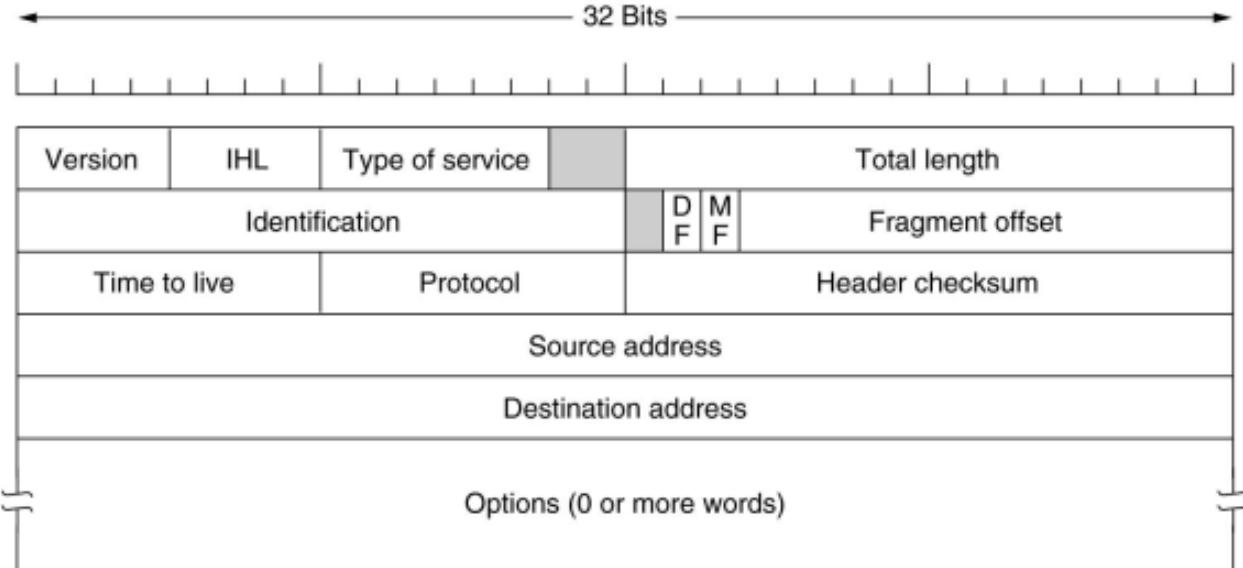
Protocolo IP

El protocolo ip es el encargado de definir las direcciones IP, redes, tablas de reenvío de paquetes y el manejo de la fragmentación de paquetes.

ip tiene dos versiones IPv4: 32bits, IPv6: 128 bits

IPv4

Datagrama IP, esta conformado por un Encabezado + Texto. Encabezado = 20Bytes + Opcional



- campo **IHL** (4b):
 - longitud del encabezado en palabras de 32b ($5 \leq \text{valor} \leq 15$).
 - 5 cuando no hay opciones.
- Campo **longitud total**: (2B) de encabezado + datos ≤ 65535 B

Campo **tipo de servicio**:

- los 2 últimos bits se usan para información de notificación de congestión (para ECN).
- Los 6 primeros bits se usan para indicar clase de servicio (p.ej. entrega rápida, transmisión libre de errores, etc.)

El campo **protocolo** (8 b) dice a cuál proceso de transporte (p.ej. TCP, UDP, etc.) entregar el paquete.

El campo **identificación** se usa para que el host de destino determine a qué paquete un fragmento pertenece.

El **campo tiempo de vida** se usa para limitar el tiempo de vida de un paquete.

- Debe decrementarse en cada salto.
- Cuando llega a cero el paquete es descartado y se manda un paquete de advertencia al host de origen.
- Esto evita que los paquetes anden dando vueltas demasiado tiempo.

El **campo suma de verificación**: se usa para detectar errores cuando el paquete viaja a lo largo de la red.

- Debe recalcularse en cada salto, porque el campo tiempo de vida siempre cambia.

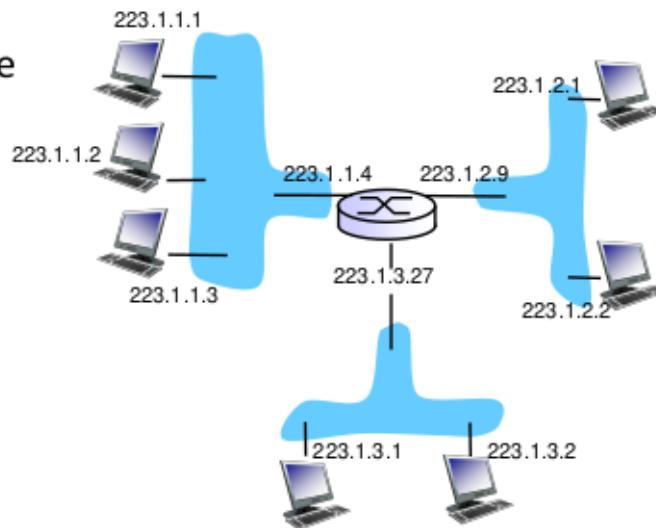
En el datagrama IP los campos dir de origen y destino son las direcciones IP de cada host

Indican numero de red y numero de maquina.

Cada host en la internet puede tener mas de un IP, una por cada red a la que esta conectada
Las direcciones van desde 0.0.0.0 a 255.255.255.255

interfaz: conexión entre host/enrutador y enlace físico.

- Un enrutador tiene muchas interfaces, una por cada línea de salida.
- Un host tiene una o dos interfaces:
 - con Ethernet cableada,
 - con inalámbrica 802.11



Cada interfaz tiene

asociada una dirección IP

$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_{1} \underbrace{00000001}_{1} \underbrace{00000001}_{1}$

Una red corresponde a un bloque contiguo del espacio de direcciones IP llamado prefijo.

- Se escriben dando la dirección IP mas baja en el bloque y la cantidad de bits usadas para la dirección de la red.

Una Subred es:

- un conjunto de interfaces de dispositivos con la MISMA parte de red de la dirección IP.
- O alternativamente máquinas que se puede alcanzar físicamente entre sí sin la necesidad de un enrutador interviniente

Receta:

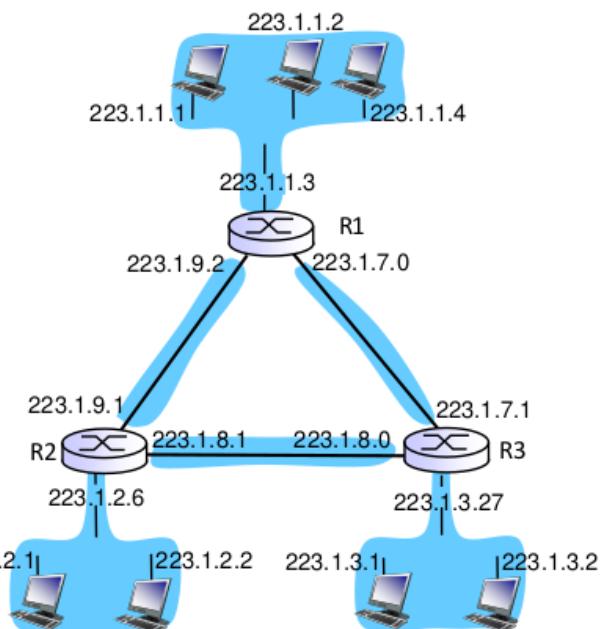
- ❖ Para determinar las subredes, desacoplar cada interfaz de su host o enrutador, creando islas de redes aisladas
- ❖ Cada red aislada se llama una **subred**
- ❖ Las subredes se indican usando prefijos

Subredes

Ejemplo:

Hay 6 subredes:

- 223.1.1.0/24
- 223.1.2.0/24
- 223.1.3.0/24
- 223.1.9.0/24 para las interfaces que conectan R1 y R2
- 223.1.8.0/24 para las interfaces que conectan R2 y R3
- 223.1.7.0/24 para las interfaces que conectan R3 y R1



Como se asignan las direcciones IP para evitar que las tablas de redirecciónamiento de los enrutadores crezca demasiado?

CIDR

La idea es alojar las direcciones IP de una red en un bloque contiguo que permite 2^k máquinas

- En todas las máquinas de la red, la parte para identificar la red es la misma.

- Se representa la red con un prefijo.

Nomenclatura: una red de /xx significa que la porción de la red tiene xx bits.

- P. ej.: una red de /20.

Una **máscara** está formada de 1s para identificar la red seguido de 0s para identificar las máquinas.

¿Cuál es la máscara de 128.208.0.0/24?

11111111 11111111 11111111 00000000

Otra forma de expresarla es: 255.255.255.0

Utilizando estas máscaras se define el enrute jerárquico, de forma que en las tablas de enrute solo se representan las redes.

- Cada entrada de tabla de enrute se extiende para darle una máscara de 32 bits.
- **Tabla de enrute** para todas las redes tiene entradas:

(dirección IP inicio subred, máscara, línea de salida.)

Como hacer la tabla de enrute para

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

- **Solución:**

- Cambridge 194.24.0.0 -> 194.24.7.255

- Para 7 necesito 3 bits (se usan junto con los 8 primeros para nº de host)
 - Máscara: 255.255.248.0 (248=1111 1000)
 - Max: 2048 hosts

- las entradas son:

- Dirección

- Máscara

- C: 11000010 00011000 00000000 00000000 11111111 11111111 11111000 00000000
 - E: 11000010 00011000 00001000 00000000 11111111 11111111 11111100 00000000
 - O: 11000010 00011000 00010000 00000000 11111111 11111111 11110000 00000000

Cual es el uso de la tabla de enrutamiento cuando llega un paquete a un router?

El enrutador extrae la dirección de destino del paquete y hace un AND bit a bit con todas las máscaras que tiene en su tabla.

De esta forma el resultado deberá coincidir con alguna de las direcciones de inicio de alguna subred de las entradas, ej: si llega una dirección IP destino se comparará con la máscara de Cambridge y luego ese resultado con la dirección base de C, luego lo mismo con E y con O. Hasta que se encuentre alguna coincidencia.

Solución: CIDR (Classless Inter Domain Routing) Cont.

- Para evitar que las tablas de enrutamiento crezcan demasiado
- se combinan varios prefijos en un prefijo único más grande (conocido como **superred**).
 - A esto se le llama **agregación de prefijos**.
- **Ejemplo:** la misma dirección IP que un enrutador trata como parte de un /22 puede ser tratada por otro enrutador como parte de un /20 más grande.
- A distintas regiones geográficas se asignan distintos espacios de direcciones. Esto se puede aprovechar en la **agregación de prefijos**:
- **Idea:** combinar prefijos de varias redes que están **en una misma región geográfica** en un prefijo para un enrutador que está en otra **región alejada**.

Una red de /c quiere decir que se le dan 2^{32-c} números IP para máquinas

- **Ejercicio:** aplicar agregación de prefijos a las 3 redes de universidades de Inglaterra (**ayuda:** ellas entran en bloque de 8192 direcciones).

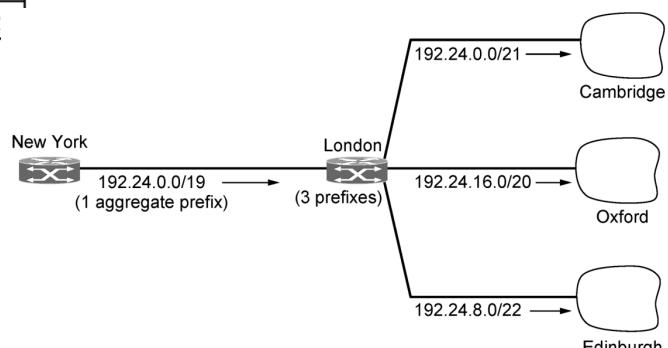
Para resolver este caso, el prefijo de toda la red será una red que tiene 8192 dir.

$$2^{13} = 8192, \text{ entonces } c = 32 - 13 = 19$$

Luego la red con agregación de prefijos será.

$$192.24.0.0/19$$

University	First address	Last address	How many	Written as
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12.0/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/2



Como podemos aumentar la cantidad de máquinas de una red de /c, arriba de las 2^{32-c} direcciones?

NAT

Solución: traducción de dirección de red (NAT).

Asignar un solo N° de IP a cada organización para el tráfico de internet.

1. Dentro de la organización cada computadora tiene una dirección IP única que se usa para el tráfico interno. (o sea, estos números IP no se usan en internet – solo adentro de la organización y pueden repetirse en distintas organizaciones)
2. Cuando un paquete sale de la organización y va al ISP, se presenta una **traducción de dirección** (de la dirección de la computadora en la organización a la dirección IP usada por la organización en internet).

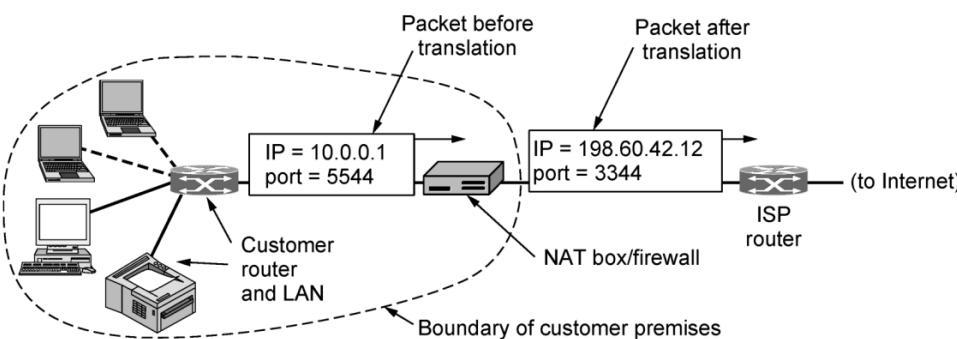


Fig. 60: Colocación y operación de la caja NAT

- Supongamos que en una organización cada máquina tiene una dirección 10.x.y.z.
- **¿Cómo hacer cuando un paquete sale de las instalaciones de la organización?**
- El paquete pasa a través de una **caja NAT** que convierte la dirección interna de origen de IP a la dirección IP de la organización.

Cada paquete TCP saliente contiene los puertos de origen y destino que sirven para identificar los procesos que usan la conexión en ambos extremos.

¿Qué pasa con el uso de puertos cuando un proceso quiere establecer una conexión TCP con un proceso remoto?

- se asocia a un puerto TCP sin usar en su máquina conocido como **puerto de origen** (indica dónde enviar mensajes entrantes de esta conexión).
- El proceso proporciona también un **puerto de destino** para decir a quién dar los mensajes en el lado remoto.

Cuando un paquete vuelve, vuelve hacia la dirección IP de la organización (o de la subred). Ahora la caja NAT debe saber a qué dirección interna mandar ese paquete.

Solución 1: Guardar asociación en la caja NAT de número

IP al puerto de origen que viene en el mensaje TCP/UDP dentro del paquete.

- Estas asociaciones se pueden guardar en una **tabla** en la caja NAT.

Esto no funciona ya que dos conexiones pueden usar el puerto de origen 5000, luego la caja nat no sabría a qué dirección con puerto de origen 5000 enviarlo.

Solución 2: distinguir entre el N° de puerto usado para identificar la máquina (o sea IPs en la red interna) y el N° de puerto usado por TCP/UDP para identificar la conexión.

- Cuando llega un paquete con puerto de origen, se busca en la tabla el IP del nodo y el N° del puerto que se usa para la conexión.

En la tabla nat los índices son números de puerto para identificar la máquina. y la entrada contiene:

-número de puerto para identificar conexión, dirección IP(interna)

Cuando un paquete llega a la caja NAT desde el ISP(internet service provider).

- se extrae el puerto del origen del encabezado TCP.
- se lo usa como índice en la tabla de traducción de NAT
- se obtiene la dirección IP interna y el puerto TCP, y se los inserta en el paquete
- se envía el paquete al enrutador interno para enviarlo a la máquina correspondiente

Tratamiento de un paquete saliente que entra en la caja NAT:

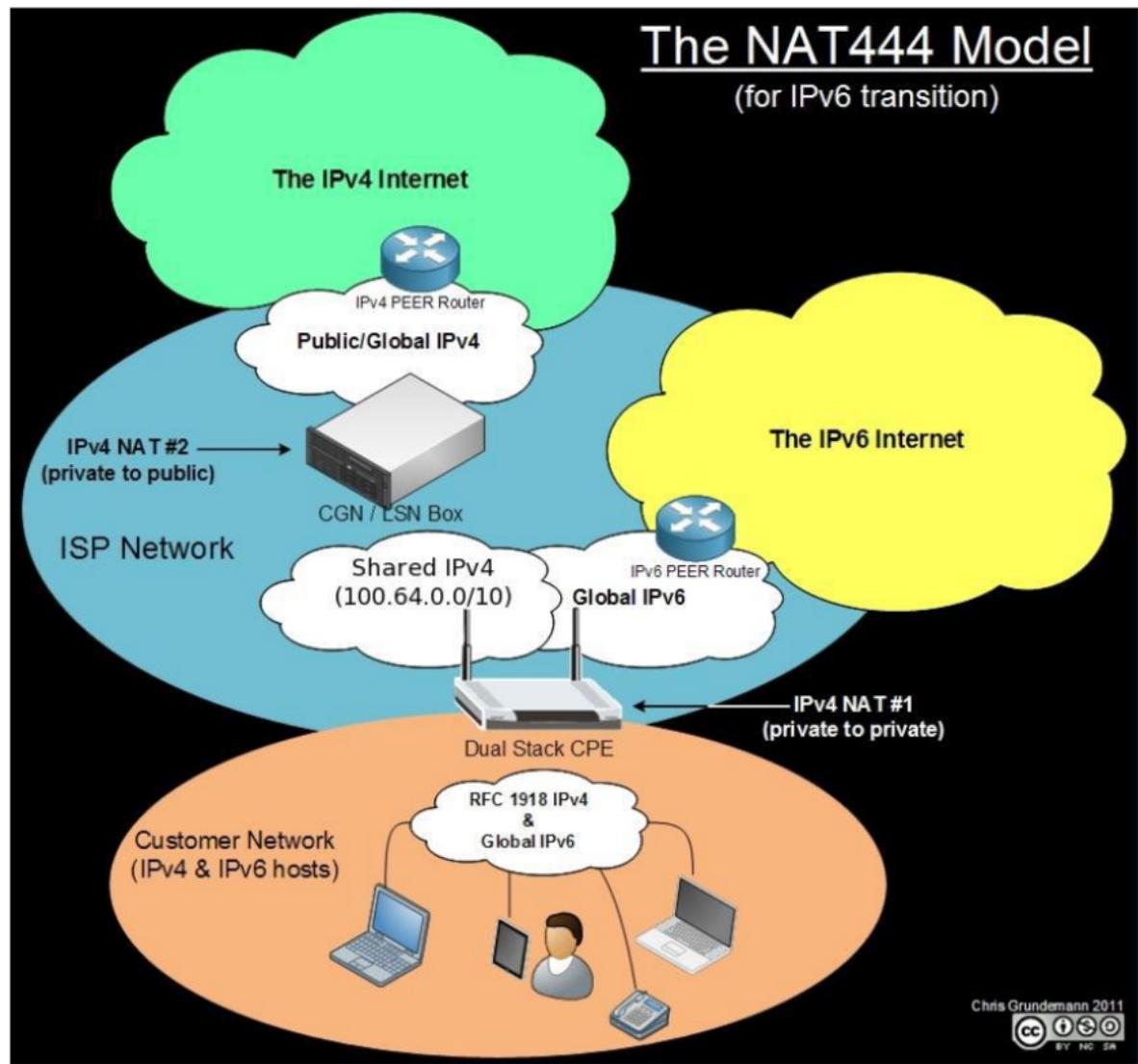
La dirección de origen 10.x.y.z se reemplaza por la verdadera dirección IP de la compañía y el campo puerto de origen TCP se reemplaza por un índice en la tabla de traducción de la caja NAT.

Con nat se puede expandir radicalmente la cantidad de máquinas y continuar usando IPv4. El problema está en que ya ninguna máquina es identificada con una única dirección IP.

Si la caja NAT se cae, todas sus conexiones TCP se destruyen.

NAT 444:

- Los proveedores de servicio de internet (PSI) también pueden tener NAT.
 - Esto hace que las direcciones IPv4 puedan racionarse más aun y durar aun más tiempo.
- Se llama NAT 444.
- El espacio de direcciones IP reservado para NAT 444 es 100.64.0.0/10 (o sea alrededor de 4000.000 de IP para ser usadas por la red del PSI)



IPv6

Como el espacio de direcciones de 32 bits ya quedo chico, entonces la idea de IPv6 es utilizar un espacio aun mayor, con menos encabezados y mas eficiente.
De esa forma el procesamiento de los encabezados no lleva tiempo y es mas eficiente.

Debido a que las redes son cada dia mas rapidas, pero la capacidad de procesamiento esta estabilizada. Entonces se deben optimizar los procesamientos de datos en los enrutadores.

Formato de datagrama IPv6:

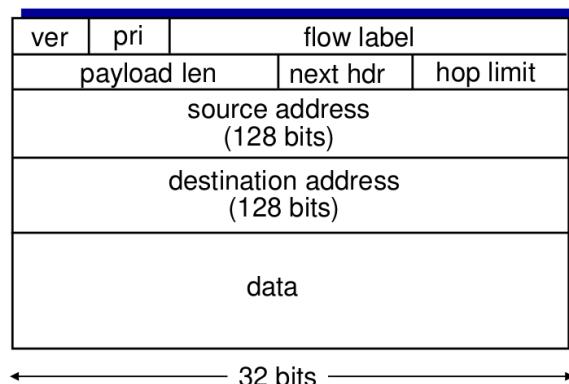
- **Encabezado de longitud fija** de 40 bytes para procesamiento más rápido de datagramas
- **Capacidad de direccionamiento expandida**: direcciones de 128 bits.
- **Etiquetado de flujos**: se etiquetan paquetes que pertenecen a un mismo flujo para los cuales el emisor requiere manejo especial.

Transmisiones de audio o video son tratados como flujo

Transferencia de archivos no.

Usuarios de alta prioridad y su trafico puede ser tratado como flujo.

La idea de etiquetar algo como un tipo de flujo, es que cuando llegue al enrutador, este sepa como tratarlo.



Etiqueta de flujo: (20 b) para identificar datagramas en el mismo "flujo".

Prioridad tiene dos usos:

- para dar prioridad a ciertos datagramas dentro de un flujo.
- para dar prioridad a datagramas de ciertas aplicaciones sobre datagramas de otras aplicaciones.

Longitud de carga útil: (16 b) número de bytes en el datagrama IPv6 luego del encabezado (de 40 B).

Límite de saltos: (8 bits) el contenido de este campo se decrementa en 1 por cada enrutador que entrega el datagrama. Si el contador alcanza 0, el datagrama se descarta.

Próximo encabezado: (8 bits) significa:

- Cuál de los 6 encabezados extensión de opciones actuales le sigue al encabezado.
- Si este encabezado es el último encabezado IP, el campo dice a cuál protocolo de transporte entregar el datagrama.
- Los encabezados de opciones tambien tienen este campo.

Direcciones IPv6:

- Son escritas como 8 grupos de 4 dígitos hexadecimales.
- Para separar los grupos se usa “:”.
- P.ej: 8000:0000:0000:0000:0123:4567:89AB:CDEF

— Optimización:

- Ceros a la izquierda de grupos pueden ser omitidos
- Grupos con 16 bits iguales a 0 pueden reemplazarse con dos “:”.

- P.ej. la dirección anterior: 8000::123:4567:89AB:CDEF

La fragmentacion o re-ensamblado no esta permitida en enrutadores intermedios.

Se remueve la suma de verificacion, para reducir el tiempo de procesamiento en cada salto.

Las opciones estan permitidas, pero fuera del encabezado. Se indican por el campo next hdr.

Los conceptos de prefijo y agregación de prefijos se usan tambien en IPv6.

- Las direcciones IPv6 se siguen asignando a interfaces.

Subredes

Una red dividida en varias partes para uso interno. Pero hacia afuera todo permanece siendo una red normal.

-Cada subred puede ser una LAN con un enrutador

-Los enrutadores de una subred conectadas a un enrutador principal

-Las subredes no son visibles fuera de la red.

Ej: una universidad tiene un enrutador principal (al que llega todo el trafico) y muchas subredes internas dispersas por las facultades.

cada subred tiene su propio enrutador conectado al enrutador principal mediante una LAN.

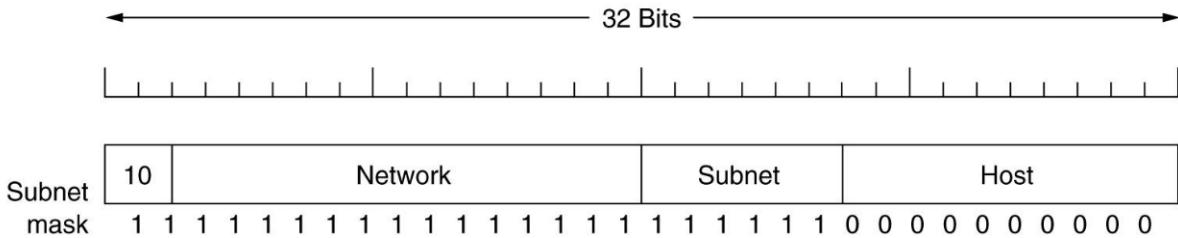
Problema: Cuando un paquete entra en el enrutador principal, ¿cómo sabe a cuál subred pasarlo?

Solución 1: tener una tabla en el enrutador principal (con tantas entradas como el tamaño de su red) que indique cuál enrutador usar para cada host.

- **Evaluación:** se requeriría una tabla muy grande en el enrutador principal y mucho mantenimiento manual conforme se agregan, movieran o eliminaran hosts.

Solución 2: algunos bits se eliminan del N° de host para crear un número de subred

- P.ej. si la universidad tiene 35 departamentos, se usa 6 bits para el número de subred y 10 bits para el número de host; lo que permite hasta 64 Ethernets, cada una con a lo más 1022 hosts.



Problema: ¿Cómo expresar subredes?

Solución: el enrutador principal usa una **máscara de subred** que indique la división entre el número de red + número de subred y el host, como se ve en la Fig. 58.

- Las máscaras de subred también se pueden escribir en notación decimal con puntos, o agregando a la dirección IP una diagonal seguida del número de bits usado para los números de red y subred.

Para el ejemplo de la Fig. 58 la máscara de subred puede escribirse como:

1111 1111. 1111 1111. 1111 1100. 0000 0000

255. 255 . 252 . 0.

- Esta máscara permite 1024 hosts
- Una notación alternativa es /22 para indicar que la máscara de subred tiene una longitud de 22 bits.

Tenemos la subred que inicia en 130.50.8.0:

1000 0010 . 0011 0010 . 0000 1000 . 0000 0000

Problema: queremos hacer la red mas grande,

Idea: deberíamos elegir otra máscara,

- por ejemplo: 255. 255. 128. 0 que albergaría $2^{15} = 32\text{ K}$ hosts.
- Esto haría que la red llegue hasta la 130.50.135.255.
- Pero escribamos la mascara en binario:

1111 1111. 1111 1111. 1000 0000 . 0000 0000

- ¡Ningún paquete que se haga AND con esta máscara me da la IP de origen de la subred! (130.50.8.0)
- Moraleja:** la cantidad máxima de hosts se da por la cantidad de 0 a la derecha del ultimo 1 en la dirección de origen:

1000 0010 . 0011 0010 . 0000 1000 . 0000 0000

OSPF

OSPF es un algoritmo de enrutamiento que funciona bien para redes IP grandes, ya que los algoritmos vistos previamente cuando las redes eran muy grandes no eran eficientes y perdían tiempo en el procesamiento.

Un **sistema autónomo** (SA) consiste de un grupo de routers que trabajan dentro de un SA. Un router trabaja dentro de un SA.

- A menudo los routers de un proveedor de servicios de internet (PSI) y los enlaces que los interconectan constituyen un SA.
- A veces un PSI divide su red en varios SA.
- Los routers dentro de un SA corren el mismo algoritmo de enrutamiento llamado **protocolo de enrutamiento intra-SA**.

Internet es un conjunto de SAs.

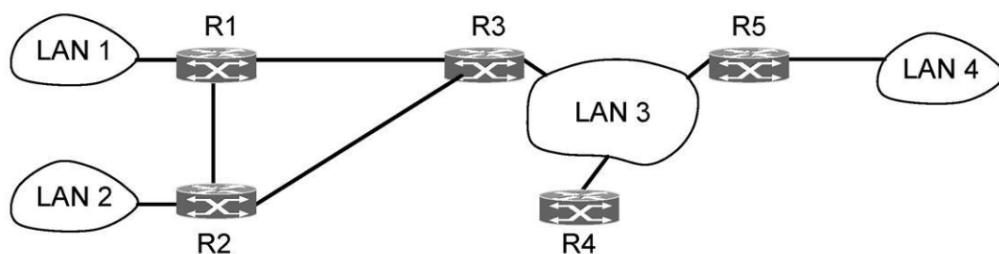
- En Internet los SA están numerados, cada uno con un número que lo identifica.

¿Por qué estudiar OSPF?

Porque OSPF introduce **mejoras interesantes** al protocolo de enrutamiento de estado de enlace:

- Es compatible con IP.
- En OSPF el modelo de grafo asociado a un SA es bastante más flexible que el usado para los protocolos de enrutamiento anteriores al considerar redes de distintos tipos.
- Para permitir SAs grandes OSPF organiza un SA como una jerarquía de niveles.
- Con OSPF para un destino se puede considerar más de una línea de salida (cuando hay más de un camino óptimo) para balancear la carga en la red.

Estas mejoras introducen **problemas nuevos** para diseñar un algoritmo de enrutamiento.



Tipos de conexiones y redes soportadas OSPF:

1. Las líneas punto a punto entre dos routers.
2. Redes de multiacceso con difusión (p.ej. la mayoría de las LAN).
3. Redes de multiacceso con muchos routers, cada uno de los cuales se puede comunicar directamente con los otros. (LAN 3 de la figura)

Para poder representar esas redes como grafos se tiene en cuenta:

- los enrutadores son los nodos
- cada arco tiene un costo de retardo
- una conexión punto-punto son dos arcos paralelos con pesos iguales o diferentes
- Una red se representan con un nodo en si (no hay una por cada dispositivo en la red)

Organización de un SA en OSPF:

- OSPF divide los SAs en **áreas numeradas**.
- *Un área puede contener varias redes adentro de ella.*
- Cada enrutador está configurado para conocer qué otros enrutadores están en su área.
- **Las áreas no se traslanan.**

Tipos de áreas en un SA:

- Hay área que es red dorsal que tiene número 0.
- Hay áreas que se conectan a la red dorsal.
 - *Se puede entrar desde un área en el SA a cualquier otra área en el SA mediante la red dorsal.*
- La topología de la red dorsal no es visible fuera de esta.

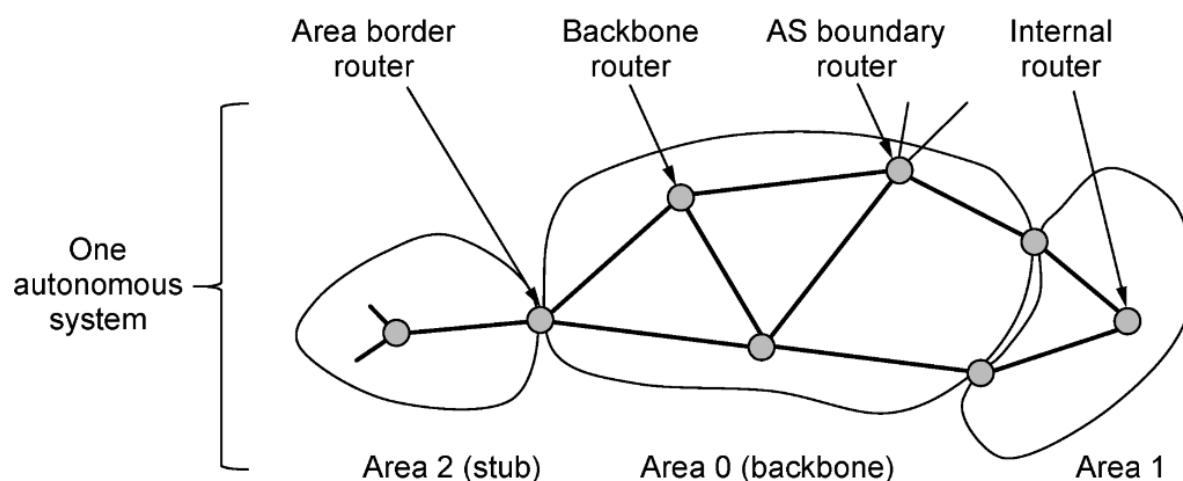
Clasificación de los enrutadores de un SA:

- **Enrutadores internos:** yacen completamente dentro de un área.
- **Enrutadores dorsales:** enrutadores en un área dorsal
- **Enrutador de borde de área (EBA).**

Es parte de una red dorsal y a la vez de una o más áreas.

Enrutador de borde de SA (EBSA).

- Inyecta en el área rutas a destinos externos en otros SA.
- Ya lo veremos con cuidado cuando estudiemos BGP.



La relación entre áreas en OSPF.

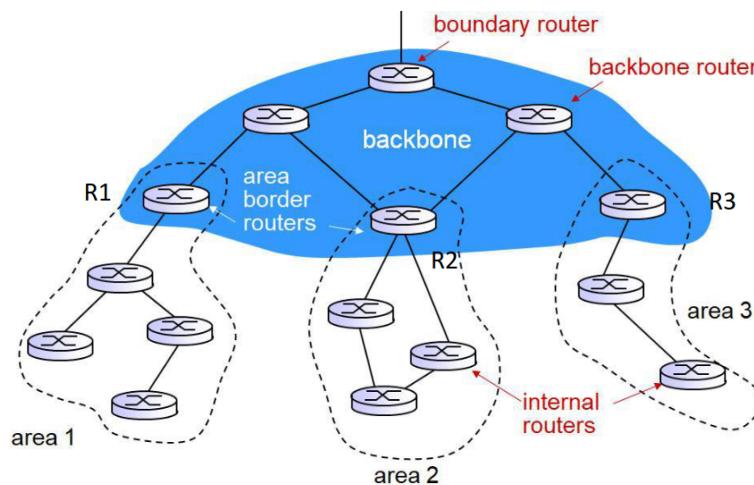
Un paquete de aviso de estado de enlace, contiene el costo de un enrutador a todos sus vecinos.

En los SA, un area no sabe como esta formada otra area, pero si sabe alguna informacion. Los enrutadores de borde de area (EBA) son los que resumen la informacion de enrutamiento aprendida de un area, y la hace disponible a los paquetes AEE que envia a otras areas.
(aviso de estado de enlace)

Problema: ¿Cómo definir la información resumida de un área no dorsal?

Solución:

- Un EBA *E* recibe **avisos de estado de enlace** de todos los enrutadores de una de sus áreas *A* y con esa información determina el **costo de alcanzar cada LAN** de *A*.
- La **información resumida de *A*** contiene el costo de alcanzar cada LAN de *A*. Este paquete es puesto por el EBA *E* en la red dorsal para que llegue a las demás áreas.



OSPF

La info que recibe un area, de una area dorsal a traves de un EBA es:
-Resumen de las areas no dorsales distintas de A
-Resumen del area dorsal

Todos los enrutadores de un area pueden aprender a alcanzar todas las redes locales en el SA.

- Sea la red dorsal de arriba y asuma que todos los arcos tienen peso 1.
- **Información resumida del área dorsal:** **Consecuencias/impacto que tiene el envío de resúmenes por un EBA para los enrutadores:**
 - Arco de R1 a R2 con costo: 2
 - Arco de R1 a R3 con costo: 4
 - Arco de R2 a R3 con costo: 2
 - Arco de R2 a R1 con costo: 2
 - Arco de R3 a R1 con costo: 4
 - Arco de R3 a R2 con costo: 2
 - Esto permite que todos los enrutadores del área dorsal aprendan el costo de alcanzar todas las redes de cada área.
 - Todos los enrutadores aprenden a alcanzar todas las redes en el SA.
 - Cada enrutador tiene una topología de su área detallada y solo conoce el costo del camino más corto a las redes en las otras áreas.