

SUMARIO:

5 - LA CAPA DE RED	343
5.1 ASPECTOS DE DISEÑO DE LA CAPA DE RED	343
5.1.1 Comutación de paquetes de almacenamiento y reenvío	344
5.1.2 Servicios proporcionados a la capa de transporte	344
5.1.3 Implementación del servicio no orientado a la conexión	345
5.1.4 Implementación del servicio orientado a la conexión	347
5.1.5 Comparación entre las subredes de circuitos virtuales y las de datagramas	348
5.2 ALGORITMOS DE ENRUTAMIENTO	350
5.2.1 Principio de optimización	352
5.2.2 Enrutamiento por la ruta más corta	353
5.2.3 Inundación	355
5.2.4 Enrutamiento por vector de distancia	357
5.2.5 Enrutamiento por estado del enlace	360
5.2.6 Enrutamiento jerárquico	366
5.2.7 Enrutamiento por difusión	368
5.2.8 Enrutamiento por multidifusión	370
5.2.9 Enrutamiento para hosts móviles	372
5.2.10 Enrutamiento en redes ad hoc	375
5.2.11 Búsqueda de nodos en redes de igual a igual	380
5.3 ALGORITMOS DE CONTROL DE CONGESTIÓN	384
5.3.1 Principios generales del control de congestión	386
5.3.2 Políticas de prevención de congestión	388
5.3.3 Control de congestión en subredes de circuitos virtuales	389
5.3.4 Control de congestión en subredes de datagramas	391
5.3.5 Desprendimiento de carga	394
5.3.6 Control de fluctuación	395
5.4 CALIDAD DE SERVICIO	397
5.4.1 Requerimientos	397
5.4.2 Técnicas para alcanzar buena calidad de servicio	398
5.4.3 Servicios integrados	409
5.4.4 Servicios diferenciados	412
5.4.5 Comutación de etiquetas y MPLS	415
5.5 INTERCONECTIVIDAD	418
5.5.1 Cómo difieren las redes	419
5.5.2 Conexión de redes	420
5.5.3 Circuitos virtuales concatenados	422
5.5.4 Interconectividad no orientada a la conexión	423
5.5.5 Entunelamiento	425
5.5.6 Enrutamiento entre redes	426
5.5.7 Fragmentación	427

SUMARIO:

5.6 LA CAPA DE RED DE INTERNET	431
5.6.1 El protocolo IP	433
5.6.2 Direcciones IP	436
5.6.3 Protocolos de Control de Internet	449
5.6.4 OSPF – Protocolos de Enrutamiento de Puerta de Enlace Interior	454
5.6.5 BGP – Protocolo de Puerta de Enlace de Frontera	459
5.6.6 Multidifusión de Internet	461
5.6.7 IP móvil	462
5.6.8 IPv6	464

5 LA CAPA DE RED

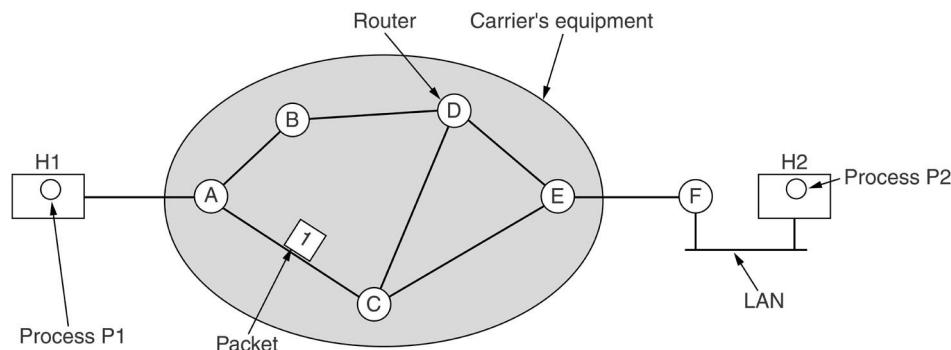
- lleva los **PAQUETES** desde el **ORIGEN** hasta el **DESTINO**, incluso cuando cada uno se encuentra en **redes diferentes y distantes**
- Esto requiere:
 - muchos **SALTOS** por **ENRUTADORES INTERMEDIOS**
 - conocer la **TOPOLOGÍA** de la **SUBRED DE COMUNICACIONES (EL GRUPO DE ENRUTADORES)**
 - elegir las **RUTAS** adecuadas **para no sobrecargar:**
 - las **líneas de comunicación**
 - ni los **enrutadores**

5.1 ASPECTOS DE DISEÑO DE LA CAPA DE RED

5.1.1 Comutación de paquetes de almacenamiento y reenvío

El **CONTEXTO** en el que operan los **PROTOCOLOS DE LA CAPA DE RED**:

- Equipo de la empresa portadora:
ENRUTADORES conectados mediante líneas de transmisión
- Equipo del cliente



Se muestran dos modalidades:

- El host H1 conectado a un enrutador de la empresa portadora mediante una línea alquilada
- El host H2 y el enrutador del cliente conectado mediante una línea alquilada

Aquí se diferencia quién ejerce el derecho de propiedad sobre el enrutador, pero en términos de construcción, software y protocolos estos enrutadores, tal vez no sean diferentes. En general la **Línea alquilada** consta de un **ENLACE DE PUNTO A PUNTO**.

Un **HOST** transmite al **ENRUTADOR** más cercano un **PAQUETE**.

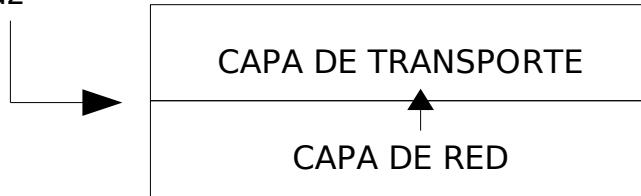
El **PAQUETE** se **almacena en el ENRUTADOR** hasta que ha llegado por completo, para poder comprobar la **SUMA DE VERIFICACIÓN**.

El **ENRUTADOR** **reenvía el PAQUETE** al **siguiente ENRUTADOR de la RUTA hacia el DESTINO**.

Se realizan tantos reenvíos del **PAQUETE** a los **ENRUTADORES** que estén en la **RUTA** como sea necesario, hasta llegar finalmente al **HOST** de **DESTINO**, donde se entrega.

5.1.2 Servicios proporcionados a la capa de transporte

Se proporcionan en la interfaz



Los servicios se diseñaron con los siguientes objetivos:

- 1.- deben ser **independientes** de la **TECNOLOGÍA del ENRUTADOR**
- 2.- La **CAPA DE TRANSPORTE** debe estar **AISLADA** de la **CANTIDAD, TIPO y TOPOLOGÍA de los ENRUTADORES**
- 3.- Las **DIRECCIONES de RED** disponibles para la **CAPA DE TRANSPORTE** deben seguir un **PLAN DE NUMERACIÓN UNIFORME**, aún a través de varias **LANs y WANs**

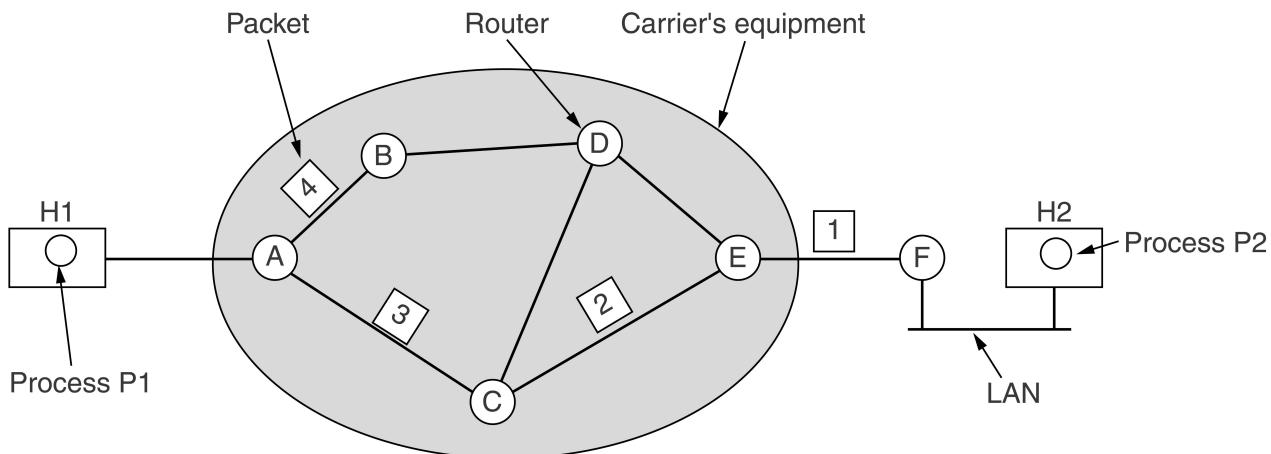
Existen **dos puntos de vista bien definidos**

LA COMUNIDAD DE INTERNET	LAS COMPAÑÍAS TELEFÓNICAS
La tarea del enrutador es mover bits de un lado a otro , y nada más.	
30 años de experiencia con una red de computadoras real y operativa	100 años de experiencia exitosa en el sistema telefónico mundial
La subred es inestable , por lo tanto, los HOST deben efectuar: - el control de errores (detección y corrección) - y el control de flujo	La subred debe proporcionar un servicio confiable
El SERVICIO DE RED: - no debe ser orientado a la conexión - debe contar tan solo con las PRIMITIVAS " SEND PACKET " y " RECEIVE PACKET " - no debe efectuar ningún ordenamiento de paquetes ni control de flujo pues los hosts ya lo realizan - cada paquete debe llevar la dirección de destino completa, pues se transporta de manera independiente de sus antecesores, si los hay	El SERVICIO DE RED: - debe ser orientado a la conexión - la CALIDAD DEL SERVICIO es un factor dominante que es muy difícil de alcanzar sin CONEXIONES en la subred, para poder brindar tráfico en tiempo REAL como la VOZ y el VIDEO
EL ejemplo es INTERNET: ofrece SERVICIOS DE CAPA DE RED NO ORIENTADO A CONEXIÓN aunque en su evolución está obteniendo propiedades asociadas a los servicios orientados a la conexión	El ejemplo es ATM

5.1.3 Implementación del servicio no orientado a la conexión

Los **PAQUETES** se colocan individualmente en la **SUBRED** y se **ENRUTAN** de manera **independiente**.

Los **PAQUETES** se conocen como **DATAGRAMAS** y la subred se conoce como **SUBRED DE DATAGRAMAS**.



A's table		C's table		E's table	
initially	later	initially	later	initially	later
A -	A -	A A	A C	A C	A C
B B	B B	B A	B D	B D	B D
C C	C C	C -	C C	C C	C C
D B	D B	D D	D D	D D	D D
E C	E B	E E	E -	E -	E -
F C	F B	F E	F F	F F	F F

Dest. Line

Un ejemplo de funcionamiento de la **SUBRED DE DATAGRAMAS**:

- El **PROCESO P1** tiene un **MENSAJE LARGO** para enviarle al **PROCESO P2**
- En el **HOST H1** el **PROCESO P1** entrega el **MENSAJE** a la **CAPA DE TRANSPORTE** y le indica que **LO ENVÍE** al **PROCESO H2** que se ejecuta en el **HOST H2**.
El **CÓDIGO DE LA CAPA DE TRANSPORTE** se ejecuta en **H1**, por lo general **dentro del SISTEMA OPERATIVO**. Este **CÓDIGO AGREGA UN ENCABEZADO DE TRANSPORTE** al **MENSAJE** y **entrega el resultado a la CAPA DE RED**, quizá OTRO PROCEDIMIENTO DENTRO DEL SISTEMA OPERATIVO.
- El tamaño del **MENSAJE**, en esta exemplificación, resulta ser 4 VECES MÁS LARGO QUE EL TAMAÑO MÁXIMO DE UN PAQUETE, por lo que **LA CAPA DE RED** lo **DIVIDE EN 4 PAQUETES: 1, 2, 3 y 4**, y **envía a cada PAQUETE** al **ENRUTADOR A**, mediante algún **PROTOCOLO PUNTO A PUNTO**, como **PPP**

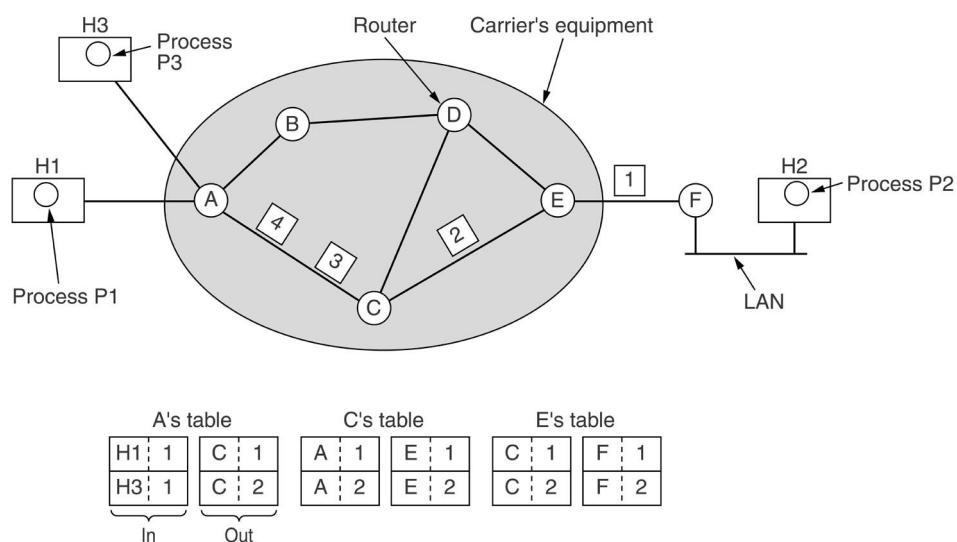
- En la **SUBRED DE DATAGRAMAS** (de la Empresa Portadora) cada **ENRUTADOR** posee una **TABLA INTERNA** que indica a **dónde enviar** los **PAQUETES** para cada **DESTINO POSIBLE**
Cada **entrada en la TABLA** es un **PAR** que consiste en un **DESTINO** y la **LÍNEA DE SALIDA** correspondiente
- A medida que los **PAQUETES** llegan al **ENRUTADOR A**, se **almacenan**, se **comprueban sus SUMAS DE VERIFICACIÓN** y se **reenvían** a los **siguientes ENRUTADORES** como se muestra en la figura, hasta que llegan al **ENRUTADOR F**, donde **SE ENCAPSULAN EN UNA TRAMA DE LA CAPA DE ENLACE** y se **envían a través de la LAN** al **HOST H2**.
El **ALGORITMO** que maneja la **TABLA** en **CADA ROUTER** y que **realiza las decisiones de ENRUTAMIENTO** se conoce como **ALGORITMO DE ENRUTAMIENTO**.

El hecho de que el PAQUETE 4 haya tomado la ruta hacia el ENRUTADOR B puede deberse a un posible problema de congestión de tráfico en alguna parte de la RUTA ACE.

5.1.4 Implementación del servicio orientado a la conexión

Antes de poder enviar cualquier **PAQUETE** de datos, es necesario **establecer una RUTA** desde el **ENRUTADOR de ORIGEN** hacia el **ENRUTADOR de DESTINO**.

Esta **CONEXIÓN** se conoce como **CIRCUITO VIRTUAL**, y la SUBRED de interconexión se llama **SUBRED DE CIRCUITOS VIRTUALES**.



El objetivo es **evitar la necesidad de elegir** una **nueva RUTA** para cada **PAQUETE** enviado.

Cuando se **ESTABLECE** una **CONEXIÓN**, se **elige una RUTA** de la máquina de **ORIGEN** a la de **DESTINO** y se la **almacena en TABLAS dentro de los ENRUTADORES**.

Esta **RUTA** se usa para todo el **TRÁFICO** que fluye a través de la **CONEXIÓN**, de manera similar a como funciona el sistema telefónico.

Cuando se **LIBERA** la **CONEXIÓN**, también se termina el **CIRCUITO VIRTUAL**.

Cada **PAQUETE** lleva un **IDENTIFICADOR** de **CIRCUITO VIRTUAL** al que pertenece, que se suele llamar **ETIQUETA**.

Para **evitar conflictos entre las distintas CONEXIONES**, los **ENRUTADORES** requieren la **CAPACIDAD de REEMPLAZAR IDENTIFICADORES DE CONEXIÓN** en los **PAQUETES SALIENTES**, esto se conoce como **CONMUTACIÓN DE ETIQUETAS**.

5.1.5 Comparación entre las subredes decircuitos virtuales y las de datagramas

ASUNTO	SUBRED DE DATAGRAMAS	SUBRED DE CIRCUITOS VIRTUALES
Configuración del circuito	No necesaria	Requerida
Direccionamiento	Cada PAQUETE contiene la DIRECCIÓN de ORIGEN y de DESTINO	Cada PAQUETE contiene un número de CIRCUITO VIRTUAL CORTO
Información de estado	Los ENRUTADORES no contienen información de estado de las CONEXIONES	Cada CIRCUITO VIRTUAL requiere ESPACIO de TABLA del ENRUTADOR POR CONEXIÓN
Enrutamiento	Cada PAQUETE se enruta de manera independiente	RTA escogida cuando se establece el CIRCUITO VIRTUAL; todos los PAQUETES siguen esta RUTA
Efecto de fallas del ENRUTADOR	Sólo se pierden los PAQUETES que estaban ENCOLADOS en el ENRUTADOR y que no se habían confirmado su recepción	Se ABORTAN todos los CIRCUITOS VIRTUALES que pasan a través del ENRUTADOR
Calidad de servicio	Difícil	Fácil se se pueden asignar suficientes recursos por adelantado para cada CIRCUITO VIRTUAL
Control de Congestión	Difícil	Fácil si pueden asignarse por adelantado suficientes recursos a cada CIRCUITO VIRTUAL

El uso de **CIRCUITOS VIRTUALES** requiere de una **FASE de CONFIGURACIÓN**, que **consume TIEMPO y RECURSOS**, pues los **PAQUETES de CONFIGURACIÓN** de la **CONEXIÓN** también **tienen que enrutarse**, ya que utilizan **DIRECCIONES DE DESTINO**, de la misma forma en que lo hacen los **DATAGRAMAS**.

5.2 ALGORITMOS DE ENRUTAMIENTO

El **ALGORITMO DE ENRUTAMIENTO** es aquella **parte del SOFTWARE de la CAPA DE RED** encargada de **DECIDIR** la **LÍNEA DE SALIDA** por la que se **TRANSMITIRÁ** un **PAQUETE de ENTRADA**.

En caso que la **SUBRED** use:

- **DATAGRAMAS**, la **DECISIÓN** debe **tomarse cada vez que llega un PAQUETE DE DATOS**
- **CIRCUITOS VIRTUALES**, la **DECISIÓN** se **toma sólo al ESTABLECERSE el CIRCUITO VIRTUAL**, y en lo sucesivo, los **restantes PAQUETES seguirán la RUTA ESTABLECIDA**
Suele denominarse **ENRUTAMIENTO DE SESIÓN**, dado que la **RUTA** permanece vigente durante toda la sesión del usuario.

Debe distinguirse que el **ENRUTAMIENTO consta de 2 PROCESOS**:

- El **MANTENIMIENTO (llenado y actualización)** de las **TABLAS de ENRUTAMIENTO**
- El **REENVÍO**, cuando el **PAQUETE LLEGA** al **ENRUTADOR** entonces **se busca en la TABLA de ENRUTAMIENTO** la **LÍNEA DE SALIDA** por la cuál se **ENVIARÁ**

Las **propiedades** que todo **ALGORITMO DE ENRUTAMIENTO** debe tener son las siguientes:

- Exactitud
- Sencillez
- Robustez:

Una vez que una **RED** principal entra en operación, se espera que **funcione continuamente durante años sin fallas** a nivel de sistema. Pero durante ese período ocurrirán **fallas de HARDWARE y de SOFTWARE** de todo tipo, ya sea en los **HOST**, los **ENRUTADORES** y en las mismas **LÍNEAS DE COMUNICACIÓN (ó CANALES INALÁMBRICOS)**. Por todo esto se exige al **ALGORITMO DE ENRUTAMIENTO** que sea **capaz de manejar los cambios de TOPOLOGÍA y TRÁFICO DE LA RED SIN REQUERIR el ABORTO** de todas las actividades en todos los **HOSTS** y el **REINICIO DE LA RED** con cada **CAÍDA del ENRUTADOR**.

- Estabilidad
- Equidad
- Optimización:

Se trata de **minimizar el RETARDO MEDIO de los PAQUETES**, pero también se desea el **AUMENTO MÁXIMO de la VELOCIDAD REAL DE TRANSPORTE de la red**

Al **reducir** el número de **SALTOS INTERMEDIOS** que tiene que dar un **PAQUETE** a través de los **ENRUTADORES** de la **SUBRED**, se está minimizando el **RETARDO** y también el **CONSUMO DE ANCHO DE BANDA**, y a su vez, se mejora la **VELOCIDAD REAL DE TRANSPORTE**.

Los **ALGORITMOS DE ENRUTAMIENTO** pueden agruparse en 2 grupos:

• **ALGORITMOS NO ADAPTATIVOS:**

No basan sus decisiones de **ENRUTAMIENTO** en mediciones o estimaciones del **TRÁFICO** y la **TOPOLOGÍA** actuales.

La **decisión** se toma **por adelantado**, fuera de línea y se carga en los **ENRUTADORES** al arrancar la **RED**.

También se conoce como **ENRUTAMIENTO ESTÁTICO**

• **ALGORITMOS ADAPTATIVOS:**

Estos **CAMBIAN** sus **decisiones de ENRUTAMIENTO** conforme a los **cambios** que van detectando **en la TOPOLOGÍA**, y en general, también **en el TRÁFICO**.

Difieren entre sí por:

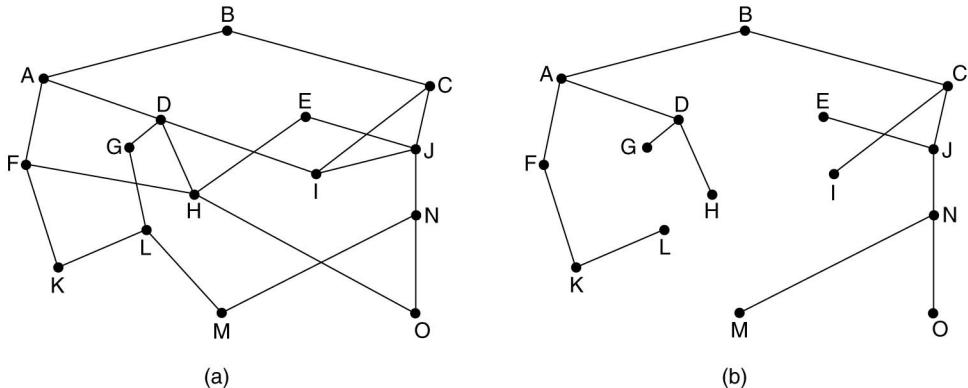
- el **lugar de dónde obtienen la información**, ya sea **localmente**, de los **ENRUTADORES** adyacentes o de **todos los ENRUTADORES**
- el **momento en que cambian sus RUTAS**, ya sea, **cada intervalo de tiempo fijo**, ó **cuando cambia la CARGA ó por cambios en la TOPOLOGÍA**
- la **MÉTRICA usada** para la optimización, ya sea, **distancia, número de saltos ó tiempo estimado de tránsito**

5.2.1 Principio de optimización

Es posible hacer un postulado general sobre las **RUTAS sin importar la TOPOLOGÍA o el TRÁFICO** de la **RED**.

El **PRINCIPIO DE OPTIMIZACIÓN** establece que si el **ENRUTADOR J** está en la **RUTA ÓPTIMA** desde el **ENRUTADOR I** hacia el **ENRUTADOR K**, entonces la **RUTA ÓPTIMA** desde el **ENRUTADOR J** hacia el **ENRUTADOR K** también está **en la misma RUTA**.

Como **consecuencia directa** de este principio, **todas las RUTAS ÓPTIMAS desde todos los ORÍGENES hacia un mismo DESTINO DADO** forman un **ÁRBOL CUYA RAÍZ es el DESTINO**, que se lo denomina **ARBOL SUMIDERO** o **ARBOL DIVERGENTE**. Este **ÁRBOL NO ES ÚNICO** ya que pueden existir otros árboles con las mismas longitudes de rutas. Todos los **ALGORITMOS DE ENRUTAMIENTO** se dedican a **DESCUBRIR** y **UTILIZAR** los **ÁRBOLES SUMIDEROS** de **TODOS LOS ENRUTADORES** presentes en la **SUBRED**.



(a) Una SUBRED

(b) ÁRBOL SUMIDERO para el ENRUTADOR B

Hay que recordar que al tratarse de un **ÁRBOL**, éste **NO CONTIENE CICLOS**, por lo que cada **PAQUETE** será entregado en un **número finito y limitado de SALTOS** entre los **ENRUTADORES** de la **SUBRED**.

El **PRINCIPIO DE OPTIMIZACIÓN** y el **ÁRBOL SUMIDERO** proporcionan parámetros de comparación para medir otros **ALGORITMOS DE ENRUTAMIENTO**.

5.2.2 Enrutamiento por la ruta más corta

Es una técnica de amplio uso, sencilla y fácil de entender. La idea es armar un **GRAFO** de la **SUBRED** en el que:

- **cada NODO** representa a **un ENRUTADOR**
- **cada ARCO** representa **una LÍNEA DE COMUNICACIÓN**

El **ALGORITMO** encuentra en el **GRAFO** la **RUTA MÁS CORTA** entre un par dado de **ENRUTADORES**.

Hay **DIVERSAS MANERAS** de **MEDIR LA LONGITUD** de una **RUTA** entre dos **ENRUTADORES**:

- por la cantidad de **SALTOS**
- por la **DISTANCIA GEOGRÁFICA** en kilómetros
- por el **RETARDO MEDIO** de **ENCOLAMIENTO** y **TRANSMISIÓN** de un **PAQUETE** estándar en cada hora transcurrida
- mediante una **FUNCIÓN DE PONDERACIÓN** que se calcula **en base a la distancia, ancho de banda, tráfico medio, costo de comunicación, longitud media de las colas, retardo medio y otros factores**

Este **ALGORITMO** se debe a **Dijkstra (1959)**.

Inicialmente todos los **NODOS** tienen la **ETIQUETA de DISTANCIA**, establecida en **INFINITO**, en forma **TENTATIVA**, pues todavía no se conocen **RUTAS**.

Una **ETIQUETA** puede ser **TENTATIVA** o **PERMANENTE**.

A medida que avanza el **ALGORITMO** y se van descubriendo las **RUTAS MÁS CORTAS** posibles **desde el ORIGEN** a cada **NODO**, su **ETIQUETA cambia a su estado PERMANENTE, y no cambia más**, para reflejar la detección de la mejor RUTA.

La **PRUEBA**, para un **NODO “i+1”** consiste en **COMPARAR**:

- Si la **SUMA** de la **ETIQUETA “i”** y la **DISTANCIA desde el NODO “i” al NODO “i+1” ES MENOR QUE** la **ETIQUETA “i+1”** **ENTONCES** tenemos una **RUTA MÁS CORTA** (que pasa por el **NODO “i”** y el **NODO “i+1”**) por lo que **RETIQUETAMOS EL NODO “i+1”** con el **RESULTADO DE ESA SUMA** en forma **TENTATIVA**
Cada vez que se **RETIQUETA UN NODO TENTATIVAMENTE** también se **almacena el NODO ANTECESOR** desde el que se hizo la prueba, para luego poder **reconstruir al final la RUTA entera**.

Esta **PRUEBA** se realiza con **TODOS LOS NODOS ADYACENTES** al **NODO DE TRABAJO**.

Tras inspeccionar **TODOS LOS NODOS ADYACENTES** al **NODO DE TRABAJO** y **CAMBIAR** las **ETIQUETAS** a **TENTATIVAS** se busca en el **GRAFO** el **NODO** etiquetado **TENTATIVAMENTE CON EL MENOR VALOR**. Este **NODO** se hace **PERMANENTE** y se convierte en el **NODO de TRABAJO** para la siguiente **RONDA**.

```

#define MAX_NODES 1024           /* número máximo de nodos */
#define INFINITY 1000000000 /* un número mayor que cualquier ruta máxima */

int n, dist[MAX_NODES][MAX_NODES]; /* dist[i][j] es la distancia de i a j */

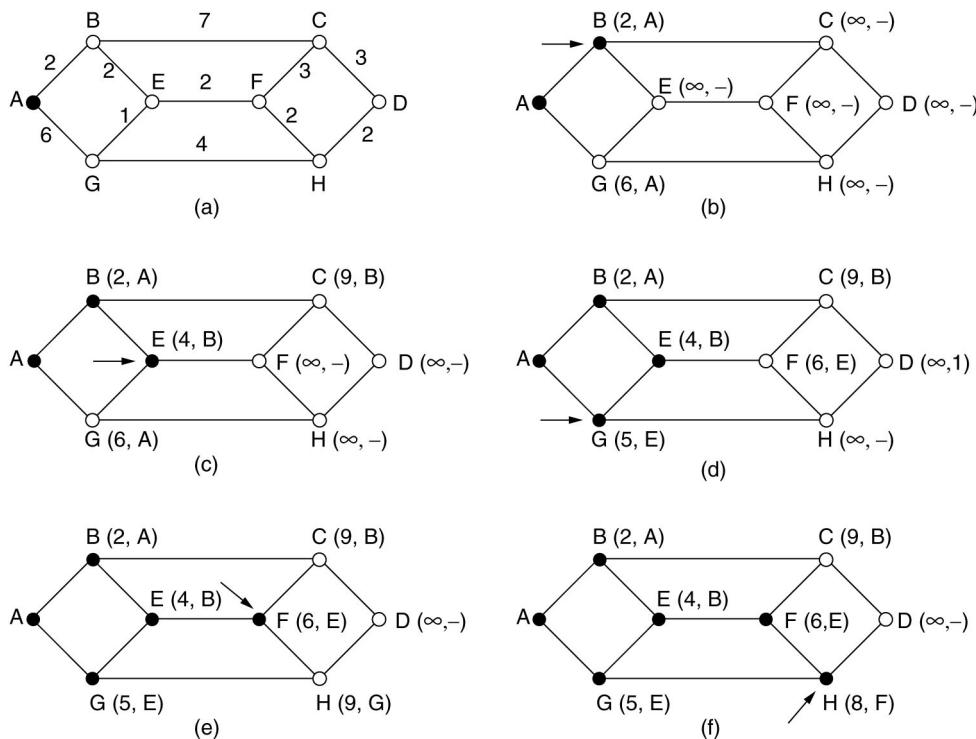
void shortest_path(int s, int t, int path[])
{ struct state {               /* la ruta con la que se está trabajando */
    int predecessor;          /* el nodo previo */
    int length;                /* longitud del origen a este nodo */
    enum {permanent, tentative} label; /* estado de la etiqueta */
} state[MAX_NODES];
int i, k, min;
struct state *p;
for (p = &state[0]; p < &state[n]; p++) { /* estado de inicialización */
    p->predecessor = -1;
    p->length = INFINITY;
    p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t;                      /* k es el nodo de trabajo inicial */
do {                         /* ¿Hay una ruta mejor desde k? */
    for (i = 0; i < n; i++)   /* este grafo tiene n nodos */
        if (dist[k][i] != 0 && state[i].label == tentative) {
            if (state[k].length + dist[k][i] < state[i].length) {
                state[i].predecessor = k;
                state[i].length = state[k].length + dist[k][i];
            }
        }
    }

/* Encuentra el nodo etiquetado tentativamente con la etiqueta menor */
k = 0; min = INFINITY;
for (i = 0; i < n; i++)
    if (state[i].label == tentative && state[i].length < min) {
        min = state[i].length;
        k = i;
    }
state[k].label = permanent;
} while (k != s);

/* Copia la ruta en el arreglo de salida */
i = 0; k = s;
do {path[i++] = k; k = state[k].predecessor;} while (k >= 0);
}

```

ALGORITMO DE DIJKSTRA para calcular la RUTA MÁS CORTA a través de un GRAFO



Los primeros cinco pasos del cálculo de la **RUTA MÁS CORTA** de A a D.
Las flechas indican el **NODO DE TRABAJO**

5.2.3 Inundación

Otro **ALGORITMO ESTÁTICO**, en el que cada **PAQUETE DE ENTRADA** se envía por **CADA UNA DE LAS LÍNEAS DE SALIDAS**, excepto aquella por la que **llegó**.

Genera **grandes cantidades de PAQUETES DUPLICADOS**. Una forma de limitarlos es integrar un **CONTADOR DE SALTOS** en el **ENCABEZADO** de cada **PAQUETE** que disminuya con cada SALTO, y el PAQUETE se descarte cuando llegue a cero.

Una **TÉCNICA** para ponerle diques a la inundación es **registrar los PAQUETES DIFUNDIDOS** para que **no se envíen por segunda vez**. Cada **ENRUTADOR de ORIGEN** debe poner un **NÚMERO DE SECUENCIA** en cada **PAQUETE** que recibe de sus **HOSTS** y además necesita una **LISTA** por **CADA ENRUTADOR** de ORIGEN que indique los números de secuencia originados en ese **ENRUTADOR**.

Una variación es la **INUNDACIÓN SELECTIVA**, en la que los **ENRUTADORES ENVÍAN** cada **PAQUETE de ENTRADA** solo por aquellas **LÍNEAS DE SALIDA** que se dirigen **APROXIMADAMENTE EN LA DIRECCIÓN CORRECTA**.

Aunque la **INUNDACIÓN no es práctica para la mayoría de las aplicaciones** tiene algunos usos:

- en **aplicaciones militares**, donde los **ENRUTADORES** pueden ser **dañados en cualquier momento**, es muy deseable su **EXCELENTE ROBUSTEZ**
- en **APLICACIONES DISTRIBUIDAS** de **BASE DE DATOS**, donde a veces es necesario actualizar concurrentemente todas las bases de datos
- en las **REDES INALÁMBRICAS**, donde las estaciones que se encuentren dentro del alcance de radio de una estación dada pueden recibir los mensajes que esta transmitan
- como **MÉTRICA** en la **comparación con otros ALGORITMOS DE ENRUTAMIENTO**. Pues la **INUNDACIÓN siempre escoge la RUTA MÁS CORTA POSIBLE**, porque **elige en PARALELO TODAS LAS RUTAS POSIBLES**. Ningún otro **ALGORITMO** puede producir un **RETARDO más CORTO, siempre y cuándo ignoremos la SOBRECARGA GENERADA por el proceso mismo de inundación**

5.2.4 Enrutamiento por vector de distancia

Se trata de un **ALGORITMO DE ENRUTAMIENTO DINÁMICO** que toma en cuenta la **CARGA ACTUAL DE LA RED**. Suele llamarse también como **ALGORITMO DE ENRUTAMIENTO de BELLMAN-FORD DISTRIBUIDO** y **ALGORITMO FORD-FULKERSON** por los investigadores que lo realizaron (Bellman 1957, y Ford y Fulkerson, 1962).

Fue el **algoritmo original de ARPANET** y también se usó **en INTERNET con el nombre RIP**.

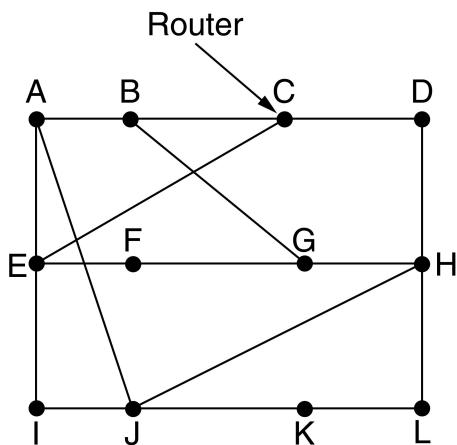
Cada **ENRUTADOR MANTIENE** una **TABLA de ENRUTAMIENTO INDIZADA por, y CONTENIENDO un REGISTRO de, CADA ENRUTADOR de la SUBRED**. Esta entrada consta de 2 partes:

- la **LÍNEA** preferida **DE SALIDA** hasta el **DESTINO**
- una **ESTIMACIÓN del TIEMPO** o de la **DISTANCIA** a ese **DESTINO (MÉTRICA)**

La **MÉTRICA** empleada puede ser:

- la **cantidad de SALTOS**
La **DISTANCIA** de cada **ENRUTADOR** a su vecino es **1**
- el **RETARDO de TIEMPO** en milisegundos
Cada **ENRUTADOR** puede medirlo mediante **PAQUETES ESPECIALES DE ECO**
- el **número total de PAQUETES ENCOLADOS** a lo largo de la **RUTA**
Cada **ENRUTADOR** examina su **COLA**.

Las **TABLAS ENRUTAMIENTO se actualizan** transcurrido un lapso de tiempo.



(a)

New estimated delay from J

To	A	I	H	K	Line
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 –
K	24	22	22	0	6 K
L	29	33	9	9	15 K

JA delay is 8 JI delay is 10 JH delay is 12 JK delay is 6

Vectors received from J's four neighbors

New routing table for J

(b)

El problema de la cuenta hasta infinito

El problema de este **ALGORITMO** es que aunque llega a la respuesta correcta, **reacciona con rapidez a las buenas noticias** (reactivaciones de **ENRUTADORES** que estaban caídos), pero con **lentitud ante las malas** (cuando los **ENRUTADORES** se apagan momentáneamente).

Las **BUENAS NOTICIAS** se difunden a razón de un **SALTO por INTERCAMBIO DE INFORMACIÓN ENTRE ROUTERS VECINOS**, por lo tanto en una **SUBRED** cuya **RUTA MAYOR** tiene una **LONGITUD de N SALTOS**, se requerirán de **N INTERCAMBIOS** para que **TODOS LOS ROUTERS ACTUALICEN** la información de las **TABLAS** y reflejen la situación sobre las **LÍNEAS** y los **ENRUTADORES REVIVIDOS**.

La razón por la cuál las malas noticias viajan con lentitud es que **NINGÚN ROUTER** jamás tiene un **valor mayor en MÁS DE UNA UNDAD** que el **MÍNIMO de TODOS SUS VECINOS**.

Todos los **ENRUTADORES elevan cuentas al infinito**, pero el número de **INTERCAMBIOS REQUERIDO** depende del **valor numérico usado para el infinito**.

Si la métrica es el **RETARDO DE TIEMPO**, se requiere de un **valor alto** para **evitar** que una **RUTA** con **RETARDO GRANDE** sea tratada como si estuviera **DESACTIVADA**

Es **prudente** hacer que el **valor para infinito** sea igual a la **RUTA MAS LARGA, MÁS UNO.**

Se han realizado varios **intentos por resolver** este **PROBLEMA DE LA CUENTA HASTA EL INFINITO**, como el **horizonte dividido con RUTAS inalcanzables del RFC 1058**, pero **ninguno funciona bien** en general.

5.2.5 Enrutamiento por estado del enlace

Se trata de otro **ALGORITMO DE ENRUTAMIENTO DINÁMICO**, que se basa en los siguientes incisos:

1. Descubrir a sus vecinos y conocer sus **DIRECCIONES DE RED**
2. Medir el **RETARDO o COSTO** para cada uno de sus vecinos
3. Construir un **PAQUETE** que **contenga todo lo aprendido**
4. Enviar este **PAQUETE** a **TODOS** los **DEMÁS ENRUTADORES**
5. Calcular la **RUTA MÁS CORTA** hacia **cada uno de los DEMÁS ENRUTADORES**

1. Conocimiento de los vecinos

Al activarse un **ENRUTADOR** su **primera acción** es **ENVIAR UN PAQUETE HELLO especial** a cada **LÍNEA PUNTO A PUNTO** para **averiguar** quiénes son sus **vecinos**.

Espera a que el **ROUTER** del otro extremo regrese una **RESPUESTA** indicando quién es.

Estos nombres deben ser globalmente únicos para ENRUTADOR de la SUBRED.

Si hay varios ROUTERS en una misma LAN, esta LAN es considerada como si se tratara de un nodo artificial nuevo que los interconecta.

2. Medición del costo de la línea

Cada **ENRUTADOR** debe **conocer el RETARDO a CADA UNO DE SUS VECINOS**.

Para ello **ENVÍA** un **PAQUETE ECHO especial** a través de la **LÍNEA** y una vez que llegue al **otro extremo**, éste **debe regresarlo** inmediatamente. Se **mide el TIEMPO DE IDA Y VUELTA y SE DIVIDE ENTRE DOS**.

Este **método supone** que los **RETARDOS SON SIMÉTRICOS**, aunque no siempre es así.

Si en la **MEDICIÓN** se **considera la CARGA**, entonces el **temporizador** debe **iniciarse cuando el PAQUETE ECHO se PONE EN LA COLA**. De esta manera si el **ENRUTADOR** debe elegir entre **DOS LÍNEAS CON EL MISMO ANCHO DE BANDA**, una **CON CARGA ALTA** y otra **SIN ELLA**, pues **considerará como RUTA MÁS CORTA LA LÍNEA SIN CARGA**. Esto resultará en un **mejor desempeño**.

Pero al suceder regularmente las **TABLAS** de los **ENRUTADORES**

pueden **OSCILAR SIN CONTROL**, lo que provocará un **ENRUTAMIENTO ERRÁTICO** y muchos **PROBLEMAS POTENCIALES**.

Si en cambio, en la **MEDICIÓN** la **CARGA** se ignora, entonces el **temporizador debe iniciarse cuando el PAQUETE ECHO alcance EL FRENTE de la COLA**.

3. Construcción de los paquetes de estado del enlace

El **PAQUETE DE ESTADO DEL ENLACE** consta de:

- la **identidad del EMISOR**
- un **número de SECUENCIA**
- una **EDAD**
- una **lista de vecinos**

La parte difícil es determinar cuándo construirlos, las posibilidades son:

- a intervalos regulares de tiempo
- cuando ocurra un evento significativo:
 - caída o reactivación de un ENRUTADOR, una LÍNEA o un VECINO
 - cambio apreciable de sus propiedades

4. Distribución de los paquetes de estado de enlace

La parte más compleja de todo el algoritmo es la distribución confiable de los **PAQUETES** de estado de enlace.

A medida que se distribuyen estos **PAQUETES** y se instalan, los **ENRUTADORES** que reciban los primeros cambiarán sus RUTAS antes. Por lo tanto, pueden producirse inconsistencias, ciclos, máquinas inalcanzables y otros problemas.

Para su distribución se emplea **INUNDACIÓN**.

Al enviar cada **PAQUETE DE ESTADO DE ENLACE**

- su **NÚMERO DE SECUENCIA** se **INCREMENTA**. Como los números de **SECUENCIA** en algún momento vuelven a comenzar desde cero, se debe utilizar un número de **SECUENCIA** de 32 bits.
- su **EDAD**, se **DECREMENTA** y a partir de allí **en forma periódica**, una vez cada segundo, para evitar pérdidas de PAQUETES y también su supervivencia durante un período indefinido. Los **PAQUETES con EDAD cero se descartan**.

Los **ENRUTADORES** llevan el **REGISTRO** de todos los valores del **PAQUETE DE ESTADO DE ENLACE: IDENTIFICADOR, SECUENCIA, EDAD y VECINOS**.

Cuando llega uno de estos **PAQUETES** al **ENRUTADOR** entra en un área de almacenamiento donde espera un tiempo breve y se **verifica sus valores** contra los **REGISTROS** de los **PAQUETES vistos hasta el momento**. Luego de su verificación satisfactoria se lo ENCOLA para su retransmisión inmediatamente.

En caso de que la verificación indique:

- NUEVO, se reenvía a través de todas las LÍNEAS excepto aquella por la que llegó (es un PAQUETE con número de SECUENCIA MAYOR que el máximo número de SECUENCIA REGISTRADO hasta el momento, y cuya EDAD es MAYOR QUE CERO) En esta instancia se decrementa en uno la EDAD y se incrementa en uno la SECUENCIA
- un duplicado en su valor de SECUENCIA, se descarta
- un PAQUETE con SECUENCIA MENOR que la MAYOR REGISTRADA hasta el momento, se RECHAZA por ser OBSOLETO

Como protección contra errores en las líneas ENRUTADOR-ENRUTADOR se confirma la recepción de todos los **PAQUETES DE ESTADO DE ENLACE**.

5. Cálculo de las nuevas rutas

Una vez que un **ENRUTADOR** ha acumulado un grupo completo de **PAQUETES DE ESTADO DE ENLACE** puede construir el **GRAFO** de la **SUBRED COMPLETA**. En este **GRAFO** cada **ENLACE** se representa **DOS VECES**, una para cada dirección.

Ahora puede ejecutar localmente el **ALGORITMO DE DIJKSTRA** para construir la **RUTA MÁS CORTA A TODOS LOS DESTINOS POSIBLES**. Los resultados pueden instalarse en las TABLAS de ENRUTAMIENTO y la operación normal puede reiniciarse.

Para una SUBRED de N ENRUTADORES, cada uno de los cuales tiene K VECINOS, la MEMORIA requerida para almacenar los datos de entrada es proporcional a K*N.

El **ENRUTAMIENTO POR ESTADO DE ENLACE** se usa ampliamente en las **REDES ACTUALES**.

Algunos PROTOCOLOS que lo usan son:

- **OSPF**: se emplea cada vez más en **INTERNET**
- **IS-IS “SISTEMA INTERMEDIO-SISTEMA INTERMEDIO”**: diseñado por **DECnet**, luego adoptado por la **ISO** para utilizarlo con su **PROTOCOLO CLNP**. Se ha ido modificando para manejar también **IP**. Se usa en varias **REDES DORSALES** como la vieja **NSFNET** y en muchos sistemas celulares digitales como **CPDP**. El **SISTEMA OPERATIVO NOVEL NETWARE** usa una **variante menor** llamada **NLSP** para el **ENRUTAMIENTO DE PAQUETES IPX**. **IS-IS** puede manejar varios **PROTOCOLOS DE RED** al mismo tiempo.

5.2.6 Enrutamiento jerárquico

La **RED** puede **CRECER** hasta el punto en que ya no es factible que cada **ENRUTADOR** tenga una entrada para cada uno de los demás **ROUTERS**.

Por lo tanto los **ENRUTADORES** se dividen en **REGIONES**, donde cada **ENRUTADOR** conoce todos los detalles para enrutar **PAQUETES** a **DESTINOS dentro de su propia REGIÓN**, pero no sabe nada al respecto sobre las otras REGIONES.

Se paga un precio, que es una **LONGITUD de RUTA MAYOR**.

Surge una pregunta:

¿Cuántos REGIONES o NIVELES debe tener una SUBRED?

Kamoun y Kleinrock (1979) descubrieron que:

- **número óptimo de niveles** para una **SUBRED** de **N ENRUTADORES** es:
 $\ln N$
- se requieren un total de: **e($\ln N$)** entradas por ENRUTADOR
- el **AUMENTO EN LA LONGITUD MEDIA EFECTIVA DE LA RUTA** es tan **pequeño** que por lo general es **aceptable**

5.2.7 Enrutamiento por difusión

El **ENVÍO SIMULTÁNEO** de un **PAQUETE** a **TODOS LOS DESTINOS** se llama **DIFUSIÓN**.

Se han propuesto varios **métodos para llevarla a cabo**:

- Un **método de DIFUSIÓN** sencillo en el que el **ORIGEN** envía un **PAQUETE DISTINTO** a **TODOS LOS DESTINOS**. Desperdicia **ANCHO DE BANDA**, requiere que el **ORIGEN** tenga una **LISTA COMPLETA** hacia **TODOS LOS DESTINOS**.
- **INUNDACIÓN**. El problema es que **genera demasiados PAQUETES** y **consume ANCHO DE BANDA**.
- **ENRUTAMIENTO MULTIDESTINO**. Cada **PAQUETE** posee una **LISTA** o un **MAPA DE BITS** que indica los **DESTINOS** deseados. Al llegar un **PAQUETE** el **ENRUTADOR** revisa **TODOS LOS DESTINOS** para determinar el **SUBGRUPO DE LÍNEAS DE SALIDA** necesario. Una **COPIA** del **PAQUETE** se genera para **cada una de esas LÍNEAS DE SALIDA**. Despues de una **cantidad suficiente de SALIOS**, cada **PAQUETE** llevará sólo un **DESTINO**.
- El **ARBOL DE EXPANSIÓN**, es un **SUBGRUPO** de la **SUBRED** que incluye a **TODOS LOS ENRUTADORES PERO NO CONTIENE CICLOS**. Cada **ENRUTADOR** debe **conocer cuáles LÍNEAS pertenecen al ARBOL DE EXPANSIÓN**, y puede **COPIAR** un **PAQUETE DE ENTRADA DIFUNDIDO EN TODAS LAS LÍNEAS** del **ARBOL DE EXPANSIÓN**, excepto aquella por la

que llegó. El **problema** es que **CADA ENRUTADOR debe tener conocimiento de algún ARBOL DE EXPANSIÓN**.

- El **REENVÍO POR RUTA INVERTIDA**. Cuando llega un **PAQUETE** difundido a un **ENRUTADOR**, éste lo **revisa para ver si llegó por la LÍNEA normalmente usada para enviar PAQUETES al ORIGEN de la DIFUSIÓN**. De ser así, el **ENRUTADOR** lo **COPIA** y **REENVÍA A TODAS LAS LÍNEAS**, excepto aquella por la que llegó. De lo contrario, el **PAQUETE** llegó por una **LÍNEA diferente de la preferida**, entonces **se descarta como probable duplicado**. La **ventaja es que no requiere que los ENRUTADORES** conozcan los **ÁRBOLES DE EXPANSIÓN**, ni tiene **LA SOBRECARGA de una LISTA DE DESTINOS O MAPA DE BITS en CADA PAQUETE**.

5.2.8 Enrutamiento por multidifusión

Algunas aplicaciones requieren que **PROCESOS** distintos trabajen juntos en **GRUPO** (procesos que implementan una **BASE DE DATOS DISTRIBUIDA**). En estos casos un **PROCESO** tiene que ser capaz de enviar un **MENSAJE** a todos los demás miembros del **GRUPO**. Por lo tanto se necesita una manera de enviar mensajes a **GRUPOS** bien definidos de tamaño grande, pero pequeños en comparación con la totalidad de la **RED**.

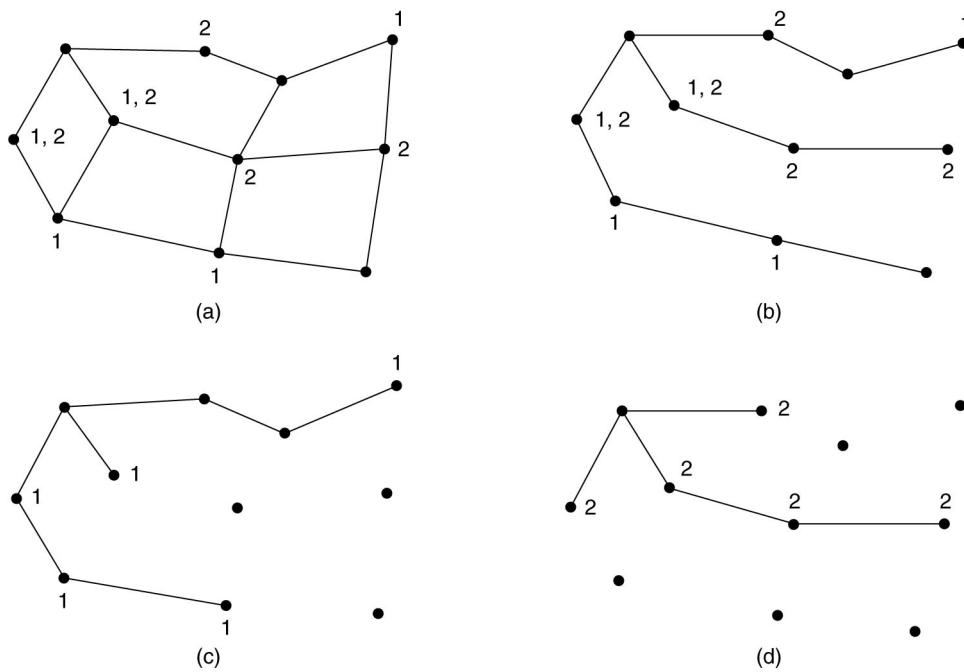
El **ENVÍO DE UN MENSAJE** a un **GRUPO** de **PROCESOS** se llama **MULTIDIFUSIÓN**.

Es necesario:

- una manera de **CREAR** y **DESTRUIR GRUPOS**
- un mecanismo para que los **PROCESOS** se **unan** a un **GRUPO** y **salgan** de ellos
- que los **PROCESOS** informen a sus **HOSTS** la **adhesión a un GRUPO**
- que los **HOSTS** informen a los **ENRUTADORES** sobre los cambios en los **MIEMBROS** de un **GRUPO**
- que los **ENRUTADORES** envíen de manera periódica la **LISTA** de sus **HOSTS** a sus **VECINOS**
- que **CADA ENRUTADOR** calcule un **ARBOL DE EXPANSIÓN** que cubra a todos los demás **ENRUTADORES** de la **SUBRED**

Algunos **ENRUTADORES** están conectados a **HOSTS** que pertenecen a uno o más **GRUPOS**.

Un **PROCESO** envía un **PAQUETE DE MULTIDIFUSIÓN** a un **GRUPO**. El primer **ENRUTADOR** examina su **ÁRBOL DE EXPANSIÓN** y lo **RECORTA**, eliminando todas **LAS LÍNEAS** que **conduzcan a HOSTS que NO SEAN MIEMBROS DEL GRUPO**.



- (a) Subred
- (b) Árbol de expansión del extremo izquierdo
- (c) Árbol de expansión del grupo 1
- (d) Árbol de expansión del grupo 2

Cuando un **ENRUTADOR** sin **HOSTS** interesados en un **GRUPO** en particular y **SIN CONEXIONES** con otros **ENRUTADORES** recibe un **MENSAJE** de **MULTIDIFUSIÓN** para ese **GRUPO**, responde con un **MENSAJE DE RECORTE (PRUNE)**, indicando al EMISOR que no envíe más **MULTIDIFUSIONES** para ese **GRUPO**.

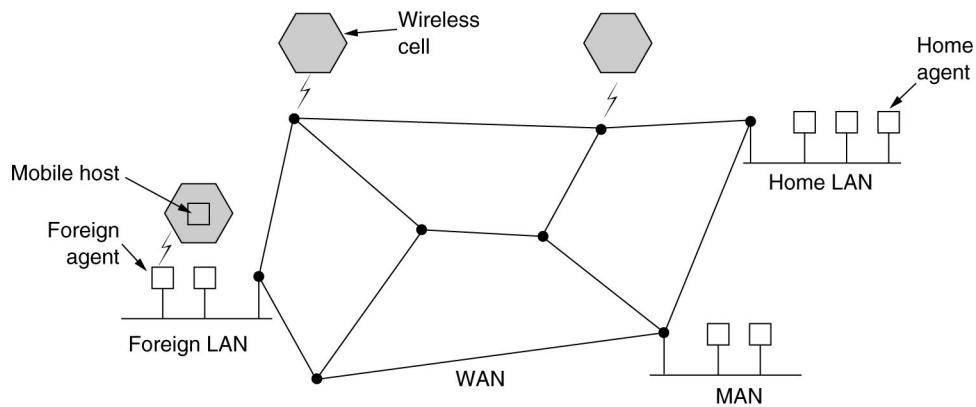
La **desventaja** del algoritmo es que **no escala bien en redes grandes**. Para una **SUBRED** con **N GRUPOS** de **PROCESOS**, cada uno con un promedio de **M MIEMBROS**, se debe almacenar, por cada GRUPO, **M ÁRBOLES DE EXPANSIÓN RECORTADOS**, lo que da un total de **M*N ÁRBOLES**.

Un **diseño alterno** utiliza **ÁRBOLES DE NÚCLEO (CORE-BASED-TREES, Ballardie y cols. 1993)** en el cuál se calcula UN SOLO ÁRBOL DE EXPANSIÓN pero cuya **RAÍZ** se localiza en el **CENTRO DE LA SUBRED**. Los **HOSTS** envían los **MENSAJES DE DIFUSIÓN HACIA EL NÚCLEO** y se **redistribuyen** mediante el **ÁRBOL**.

5.2.9 Enrutamiento para hosts móviles

Para ENRUTAR un PAQUETE a un HOST MÓVIL, la RED primero tiene que encontrarlo.

Una **WAN** consiste de **ENRUTADORES** y **HOSTS**. Conectados a la **WAN** hay varias **LANs, MANs y CELDAS INALÁMBRICAS**.



Los HOSTS:

- que nunca se mueven son estacionarios y se conectan a la SUBRED mediante cables de cobre o fibra óptica
- móviles** hacen su cómputo en movimiento y necesitan **MANTENER SUS CONEXIONES MIENTRAS SE TRASLADAN DE UN LADO A OTRO**
- tienen una **LOCALIDAD BASE** que nunca cambia
- tienen una **DIRECCIÓN BASE PERMANENTE** que puede servir para determinar su **LOCALIDAD BASE**

La **meta** es **posibilitar el envío de PAQUETES a HOSTS MÓVILES** usando su **DIRECCIÓN BASE, eficientemente** y hacia cualquier lugar en el que puedan estar.

Para ello el **MUNDO** se **DIVIDE GEOGRÁFICAMENTE EN ÁREAS**.

Cada **ÁREA** tiene:

- uno o más **AGENTES FORÁNEOS: PROCESOS** que se encargan del **REGISTRO** de **TODOS LOS HOSTS MÓVILES** que visitan el **ÁREA**
- un **AGENTE DE BASE**: que lleva el **REGISTRO** de **TODOS los HOSTS** cuya **BASE** está en el **ÁREA**, pero que actualmente están visitando otra **ÁREA**

Cuando un **HOSTS INGRESA** en un **ÁREA**, al conectarse ya sea a la **LAN** o la **CELDA INALÁMBRICA**, se debe **REGISTRAR** con el **AGENTE FORÁNEO**. El procedimiento de **REGISTRO** consta de:

1. Cada **AGENTE FORÁNEO DIFUNDE** un **PAQUETE** que **ANUNCIA SU EXISTENCIA**. Los **HOSTS MÓVILES ESPERAN** uno de estos **PAQUETES** hasta que llega, o emiten un **PAQUETE** que diga: “**¿HAY ALGÚN AGENTE FORÁNEO POR AHÍ?**”
2. El **HOST MÓVIL** se **REGISTRA** con el **AGENTE FORÁNEO** dando **SU DIRECCIÓN BASE**, su **DIRECCIÓN DE CAPA DE ENLACE DE DATOS** y cierta **INFORMACIÓN DE SEGURIDAD**

3. El **AGENTE FORÁNEO** contacta al **AGENTE DE BASE** del **HOST MÓVIL** y le dice: “**UNO DE TUS HOSTS ESTÁ POR AQUÍ**”. Este **MENSAJE** contiene la **DIRECCIÓN ACTUAL DE LA CAPA DE RED** del **AGENTE FORÁNEO** e **INFORMACIÓN DE SEGURIDAD**
4. El **AGENTE DE BASE** examina la **INFORMACIÓN DE SEGURIDAD** que contiene una **MARCA DE TIEMPO** para comprobar que fue generada en los últimos segundos. Si está conforme indica al **AGENTE FORÁNEO** que proceda
5. El **AGENTE FORÁNEO** recibe la **CONFIRMACIÓN DE RECEPCIÓN** del **AGENTE DE BASE**, ingresa una **ENTRADA EN SUS TABLAS** en informa al **HOST MÓVIL** que ya ha sido registrado

Cuando un **HOST SALE** de un **ÁREA** debe **ANUNCIARSE** para permitir que se **BORRE el REGISTRO**.

Cuando un **PAQUETE** se **ENVÍA** a un **HOST MÓVIL**, se lo **ENRUTA** a la **LAN BASE del HOST**.

5.2.10 Enrutamiento en redes ad hoc

Se trata del **caso extremo** en que los **ENRUTADORES** mismos son **MÓVILES**:

- Vehículos militares
- Flota de barcos en el mar
- trabajadores de emergencia en un área sin infraestructura
- una reunión de personas con computadoras portátiles en un área que no cuenta con **802.11**

En todos los casos cada **NODO** consiste de **un ENRUTADOR y un HOST** generalmente **en la misma computadora**.

Las redes de **NODOS** cercanos se conocen como **REDES AD HOC** o **MANETs (Redes AD HOC MÓVILES)**. En estas redes los **ENRUTADORES** pueden ir y venir o aparecer en nuevos lugares en cualquier momento, la **TOPOLOGÍA** puede cambiar todo el tiempo y la validez de las **RUTAS** puede cambiar en cualquier instante.

El **ALGORITMO DE ENRUTAMIENTO AODV “VECTOR DISTANCIA AD HOC BAJO DEMANDA”** está adaptado para funcionar en **entornos móviles** y **toma en cuenta el ANCHO DE BANDA** y la **DURACIÓN DE LAS BATERÍAS**. Se le denomina **BAJO DEMANDA** porque **determina una RUTA a algún DESTINO** sólo **cuando alguien desea enviar un PAQUETE a ese DESTINO**.

Descubrimiento de la ruta

Los **NODOS** se conectan (tienen una **ARCO** entre ellos en el **GRAFO**) si se pueden comunicar de manera directa, sin obstáculos, mediante sus radios.

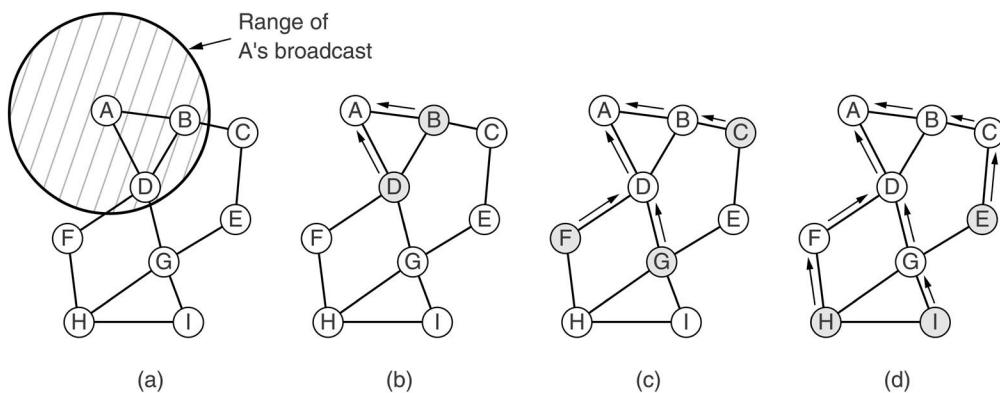
Se mantiene una **TABLA en CADA NODO**, codificada por **DESTINO**, con información acerca de ese **DESTINO** (vecino al que debe enviar los **PAQUETES** para llegar al **DESTINO**).

Para localizar al NODO DESTINO, el NODO ORIGEN construye un PAQUETE ESPECIAL DE SOLICITUD DE RUTA (ROUTE REQUEST) y lo DIFUNDE. El PAQUETE llega a los NODOS INTERMEDIOS que están en el rango de alcance.

Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count
----------------	------------	---------------------	-------------------	------------------	-----------

El **PAQUETE ESPECIAL DE SOLICITUD DE RUTA (ROUTE REQUEST)** consta de los siguientes CAMPOS:

- **DIRECCIÓN DE ORIGEN**, en general una **DIRECCIÓN IP**
- **DIRECCIÓN DE DESTINO**, por lo general también una **DIRECCIÓN IP**
- **IDENTIFICADOR DE SOLICITUD**, un **CONTADOR** local que **se mantiene por separado en cada NODO** y **se INCREMENTA cada vez que se DIFUNDE** uno de estos **PAQUETES**
- **CONTADOR DE SECUENCIA del NODO DE ORIGEN**
- **CONTADOR DE SECUENCIA DESTINO**, es el **VALOR MÁS RECIENTE** de **NÚMERO DE SECUENCIA DEL NODO DESTINO** que **HA VISTO EL NODO DE ORIGEN**



(a) Alcance de la DIFUSIÓN del NODO A

(b) Despues de que B y D reciben la DIFUSIÓN de A

(c) Despues de que C, F y G reciben la DIFUSIÓN de A

(d) Despues de que E, H e I reciben la DIFUSIÓN de A.
Los NODOS sombreados son nuevos receptores.
Las flechas muestran las RUTAS INVERTIDAS

Al llegar el **PAQUETE ESPECIAL DE SOLICITUD DE RUTA** a cada **NODO INTERMEDIO** se lo **PROCESA** de acuerdo a los siguientes 3 pasos:

1. El **PAR (DIRECCIÓN DE ORIGEN, IDENTIFICADOR DE SOLICITUD)** se busca en la **TABLA** local para ver si existe, en cuyo caso significa que la **SOLICITUD** ya **HABÍA SIDO PROCESADA, o sea** es un **DUPLICADO** entonces se detiene el procesamiento. **Si NO ES UN DUPLICADO, el PAR se introduce en la TABLA y se continúa el procesamiento**
2. El **NODO RECEPTOR** busca en su **TABLA** el **DESTINO**. Si se conoce una **RUTA RECIENTE al DESTINO** (el **NÚMERO DE SECUENCIA DE DESTINO** guardado en la **TABLA** debe ser **MAYOR o IGUAL QUE** el **NÚMERO DE SECUENCIA** del **PAQUETE DE SOLICITUD DE RUTA**) se **REGRESA** un **PAQUETE DE RESPUESTA DE RUTA (ROUTE REPLY)** al **NODO ORIGEN**

Source address	Destination address	Destination sequence #	Hop count	Lifetime
----------------	---------------------	------------------------	-----------	----------

3. El **RECEPTOR INCREMENTA** el **CAMPO CUENTA DE SALTOS** y vuelve a **DIFUNDIR** el **PAQUETE ESPECIAL DE SOLICITUD DE RUTA**. También extrae los datos del PAQUETE y los almacena como una **ENTRADA NUEVA EN LA TABLA DE RUTAS INVERTIDAS** para que la **RESPUESTA** pueda regresar posteriormente al **NODO ORIGEN**. También se inicia UN TEMPORIZADOR para la NUEVA ENTRADA en la TABLA RUTAS INVERTIDAS. Si expira, la entrada se borra.

En algún momento el **PAQUETE ESPECIAL DE SOLICITUD DE RUTA** finalmente alcanza un **NODO** que sabe dónde está el **NODO DESTINO**, puede ser el mismo **NODO DESTINO**. En respuesta a la **SOLICITUD ENTRANTE** el **NODO DESTINO** construye un **PAQUETE DE RESPUESTA DE RUTA**. La **DIRECCIÓN DE ORIGEN, DIRECCIÓN DE DESTINO** y **CUENTA DE SALTOS** se **COPIAN** de la **SOLICITUD ENTRANTE**, pero el **NÚMERO DE SECUENCIA DE DESTINO** se toma de su **CONTADOR** en la **MEMORIA DEL NODO DESTINO**. El **CAMPO CUENTA DE SALTOS** se **ESTABLECE** a **CERO**. El **CAMPO TIEMPO DE VIDA** controla cuánto tiempo es válida la **RUTA**.

El **PAQUETE DE RESPUESTA DE RUTA** se difunde únicamente al **NODO INTERMEDIO** de donde proviene la **SOLICITUD DE RUTA**. Despues sigue la **RUTA INVERTIDA** y finalmente llega al **NODO ORIGEN**.

En su recorrido por la **RUTA INVERTIDA** el **PAQUETE DE RESPUESTA DE RUTA** se inspecciona en **CADA NODO INTERMEDIO**. Se introduce en la **TABLA de ENRUTAMIENTO** del **NODO INTERMEDIO** como una **RUTA AL NODO DESTINO** si cumple una o más de las siguientes tres condiciones:

1. **No se conoce una RUTA al DESTINO**

2. El **NÚMERO DE SECUENCIA** para **DESTINO** en el **PAQUETE DE RESPUESTA DE RUTA** ES MAYOR QUE el **VALOR** en la **TABLA DE ENRUTAMIENTO**

3. Los **NÚMEROS DE SECUENCIA SON IGUALES** pero la nueva **RUTA ES MÁS CORTA**

Así todos los **NODOS INTERMEDIOS** aprenden la **RUTA HACIA EL NODO DESTINO**.

El algoritmo genera muchas DIFUSIONES, que se pueden reducir de acuerdo a lo siguiente:

- El CAMPO TIEMPO DE VIDA del PAQUETE se INICIALIZA al DIÁMETRO ESPERADO de la RED y se DECREMENTA en CADA SALTO. Si llega a CERO se descarta
- El proceso de descubrimiento se modifica. Para localizar a un DESTINO el EMISOR DIFUNDE un PAQUETE DE SOLICITUD DE RUTA con el CAMPO TIEMPO DE VIDA establecido en 1. Si después de un tiempo no se obtiene respuesta, se envía otro PAQUETE pero con el CAMPO TIEMPO DE VIDA valorizado en 2. Los intentos subsiguientes continuarán incrementando el CAMPO TIEMPO DE VIDA causando que las búsquedas se realicen en forma de anillos cada vez más grandes.

Mantenimiento de rutas

Cada NODO:

- DIFUNDE de manera periódica un mensaje de saludo (HELLO).
- espera que cada uno de sus NODOS VECINOS RESPONDA
 - Si NO recibe respuesta:
 - comprueba que el NODO VECINO SE HA MOVIDO del ALCANCE y ya NO ESTÁ CONECTADO
 - VERIFICA su TABLA de ENRUTAMIENTO para ver cuáles DESTINOS tienen RUTAS en las que se incluya al NODO VECINO AHORA INALCANZABLE, y las elimina
 - Informa a los NODOS VECINOS ACTIVOS que sus RUTAS a través de ese NODO INACTIVO ahora es inválida y debe eliminarse de sus TABLAS DE ENRUTAMIENTO
 - para cada DESTINO POSIBLE mantiene un REGISTRO de sus VECINOS ACTIVOS que le han enviado un PAQUETE para ESE DESTINO, durante el último lapso de tiempo, en una TABLA DE ENRUTAMIENTO CODIFICADA POR DESTINO

Esta información se usa para ELIMINAR RUTAS que ya no funcionan.

Los **VECINOS ACTIVOS** avisan del **NODO PERDIDO** a sus **VECINOS ACTIVOS** y así sucesivamente, hasta que las **RUTAS** que dependen del **NODO PERDIDO** se **ELIMINAN** de **TODAS LAS TABLAS DE ENRUTAMIENTO**.

Hay que destacar que como en el **ALGORITMO AODV**, los **NODOS** no envían **DIFUSIONES** periódicas que contengan su **TABLA DE ENRUTAMIENTO COMPLETA**, se **ahorra ANCHO DE BANDA** y **DURACIÓN DE BATERÍAS**.

AODV también es capaz de realizar **ENRUTAMIENTO DE DIFUSIÓN** y **MULTIDIFUSIÓN**.

5.2.11 Búsqueda de nodos en redes de igual a igual

La primera aplicación que usó esta tecnología de **IGUAL A IGUAL** fue **NAPSTER**.

Los sistemas de **IGUAL A IGUAL** son totalmente **DISTRIBUIDOS**, todos sus nodos son simétricos y no hay control central ni jerarquía.

Cada usuario tiene algo de información que podría ser útil para otros usuarios. Cada elemento de información se identifica con una **CADENA ASCII**.

Al aumentar demasiado el número de usuarios, estos no podrán conocerse entre sí y no sabrán en dónde buscar lo que desean.

El problema es qué debe hacer un usuario para encontrar un **NODO** que contenga la información que se está buscando.

Una **BASE DE DATOS CENTRALIZADA** no es una solución factible.

Un usuario potencial solo conoce la **CADENA ASCII** que denota a la información deseada y desea averiguar si una o más personas tienen copias, y de ser así cuáles son sus **DIRECCIONES IP**.

Entre los algoritmos que se han presentado, hay uno que se denomina **SISTEMA CHORD**.

Consiste de **N USUARIOS** participantes, cada uno de los cuales contiene algunos **REGISTROS** almacenados y además está preparado para almacenar **BITS Y FRAGMENTOS DEL ÍNDICE** para que los restantes **USUARIOS** los utilicen.

A la **DIRECCIÓN IP** de cada **NODO** se le aplica **UNA FUCIÓN DE HASH**, llamada **sha-1** que produce un **número completamente aleatorio de 160 bits**, que se empleará como **IDENTIFICADOR DE NODO**.

Esta misma función de hash se aplica a las **CADENAS ASCII** para obtener lo que llamaremos **CLAVE**.

Se emplea otra función **sucesor(K)**, que devuelve el **IDENTIFICADOR DE NODO** del primer **NODO REAL** que sigue a **K**.

Una búsqueda típica luce como: **(CADENA ASCII, MI-DIRECCIÓN-IP)** Para poder encontrar la **DIRECCIÓN IP** del **NODO** que corresponde a la **CADENA ASCII** buscada, cada **NODO** debe mantener ciertas estructuras de datos administrativas. Para ello se utiliza la **TABLA FINGER**.

La **TABLA FINGER** contiene tantas entradas como **NODOS** presentes.
Cada una de estas entradas tiene **2 CAMPOS**:

- **inicio**
- **DIRECCIÓN IP de sucesor(inicio)**

Los valores de estos **CAMPOS** para la **ENTRADA i en el NODO k** son:

- **inicio = $k + 2^i$ (módulo 2^m)**
- **Dirección IP de sucesor(inicio[i])**

La TABLA FINGER SE VA MODIFICANDO A MEDIDA QUE VAN INGRESANDO NUEVOS NODOS, o CUANDO UN NODO SE SEPARA.

5.3 ALGORITMOS DE CONTROL DE CONGESTIÓN

La **CONGESTIÓN** sucede cuando **hay demasiados PAQUETES EN LA SUBRED** (o **en parte de ella**), y **ocurre una DEGRADACIÓN DEL DESEMPEÑO**.

A medida que **AUMENTA EL TRÁFICO**, los **ENRUTADORES** ya no pueden manejarlo y comienzan a **PERDER PAQUETES**. Con **EXCESIVO TRÁFICO**, el **DESEMPEÑO** se **DESPLOMA** y CASI NO HAY ENTREGA DE PAQUETES.

Esto **puede ocurrir por varias razones**:

- Cuando de manera repentina al llegar PAQUETES POR 3 o 4 **LÍNEAS DE ENTRADA** y **todos necesitan la MISMA LÍNEA DE SALIDA**, se generará una **COLA**. Si **no hay SUFICIENTE MEMORIA** para almacenar los **PAQUETES**, algunos de ellos se **PERDERÁN**. **Nagle(1978) descubrió que la adición de más memoria no soluciona el problema**, al contrario lo empeora, debido a que los **TEMPORIZADORES EXPIRAN Y SE GENERAN DUPLICADOS**
- Si las **CPUs de los ENRUTADORES SON LENTAS** para realizar las tareas administrativas:
 - **BÚFERES DE ENCOLAMIENTO**
 - **ACTUALIZACIÓN DE TABLAS**
- Cuando **LAS LÍNEAS DE COMUNICACIÓN POSEE POCO ANCHO DE BANDA**

El **CONTROL DE CONGESTIÓN** se ocupa de asegurar que la **SUBRED** sea capaz de **TRANSPORTAR el TRÁFICO OFRECIDO**. Es un **asunto GLOBAL**, en el que interviene el comportamiento de **TODOS LOS HOSTS, TODOS LOS ENRUTADORES, EL PROCESO DE ALMACENAMIENTO Y REENVÍO DENTRO DE LOS ENRUTADORES y TODOS LOS DEMÁS FACTORES** que tienden a disminuir la **CAPACIDAD de la SUBRED**.

No debe confundirse **CONTROL DE CONGESTIÓN** con el

CONTROL DE FLUJO, que se relaciona con el **TRÁFICO PUNTO A PUNTO ENTRE UN EMISOR Y UN RECEPTOR** La tarea del **CONTROL DE FLUJO** es asegurar que **UN EMISOR RÁPIDO** no pueda transmitir datos de manera continua a una **VELOCIDAD MAYOR QUE** la que **PUEDE ABSORVER EL RECEPTOR** Es evidente la necesidad de una **RETROALIMENTACIÓN DIRECTA DEL RECEPTOR AL EMISOR**, para indicar al **EMISOR** cómo van las cosas en el otro lado

La **razón por la que se confunden** estos **CONCEPTOS**, es que **ALGUNOS ALGORITMOS DE CONTROL DE CONGESTIÓN** operan **ENVIANDO MENSAJES DE REGESO** a **VARIOS ORÍGENES**, indicándoles que **REDUZCAN LA VELOCIDAD CUANDO LA SUBRED SE METE EN PROBLEMAS**.

5.3.1 Principios generales del control de congestión

El análisis de las REDES DE COMPUTADORAS de acuerdo a la TEORÍA DE CONTROL nos proporciona DOS GRUPOS DE SOLUCIONES POSIBLES:

- **SOLUCIONES DE CICLO ABIERTO:** implica la toma de **decisiones independientemente del ESTADO ACTUAL DE LA SUBRED**
 - decidir cuándo aceptar TRÁFICO NUEVO
 - decidir cuándo DESCARTAR PAQUETES, y cuáles
 - tomar decisiones de calendarización en varios puntos de la RED
- **SOLUCIONES DE CICLO CERRADO:** se basan en un **CONCEPTO DE UN CICLO DE RETROALIMENTACIÓN**, el método tiene 3 partes:
 1. Monitorear el sistema para detectar cuándo y dónde ocurren CONGESTIONES
 2. Pasar esta información a lugares en los que puedan llevarse a cabo acciones
 3. Ajustar la operación del sistema para corregir el problema

Se **pueden utilizar VARIAS MÉTRICAS** para realizar el **MONITOREO**:

- **PORCENTAJE DE PAQUETES DESCARTADOS POR FALTA DE ESPACIO EN BÚFER**
- **LONGITUD PROMEDIO DE LAS COLAS**
- **CANTIDAD DE PAQUETES** para los cuales **TERMINA UN TEMPORIZADOR y SE TRANSMITE UNA NUEVA CUENTA**
- **RETARDO PROMEDIO DE LOS PAQUETES**
- **DESVIACIÓN ESTÁNDAR DEL RETARDO DE PAQUETES**

Hay también **varias opciones** para la **transferencia de la información relativa** a la **CONGESTIÓN** desde el **punto en que se detecta** hasta el **punto en que puede hacerse algo al respecto**:

- El **ENRUTADOR** que **DETECTE** la **CONGESTIÓN ENVÍE** un **PAQUETE** al **ORIGEN (u ORÍGENES)** del **TRÁFICO**. Pero estos **PAQUETES adicionales AUMENTAN LA CARGA**

- Cada **PAQUETE** contemple la reservación de **uno de sus bits**, o un **CAMPO**, que **sirva para que los ENRUTADORES** lo llenen **cuando la CONGESTIÓN rebase algún umbral**, y de esta manera **poder avisar a sus VECINOS**
- Los **HOSTS** o **ENRUTADORES ENVÍEN DE MANERA PERIÓDICA PAQUETES DE SONDEO** para **VERIFICAR** si hay **CONGESTIÓN**

Yang y Reddy (1995) han desarrollado una **taxonomía de los ALGORITMOS DE CONTROL DE CONGESTIÓN**:

- **DE CICLO ABIERTO:**
 - **QUE ACTÚAN EN EL ORIGEN**
 - **QUE ACTÚAN EN EL DESTINO**
- **DE CICLO CERRADO:**
 - **RETROALIMENTACIÓN EXPLÍCITA:**
Regresan **PAQUETES** desde el **PUNTO DE CONGESTIÓN** al **ORIGEN**
 - **RETROALIMENTACIÓN IMPLÍCITA:**
El **ORIGEN DEDUCE LA EXISTENCIA** de una **CONGESTIÓN** haciendo observaciones locales

Ante la presencia de CONGESTIÓN se puede optar por:

- **AUMENTAR LOS RECURSOS**, siempre que sea posible
- **DISMINUIR LA CARGA**, a un mínimo indispensable

5.3.2 Políticas de prevención de congestión

Se trata de reducir la **CONGESTIÓN** desde el inicio mismo

CAPA	POLÍTICAS
TRANSPORTE	<p><u>Política de retransmisión</u></p> <p><u>Política de almacenamiento en caché de paquetes fuera de orden</u></p> <p><u>Política de confirmación de recepción</u></p> <p><u>Política de control de flujo</u></p> <p><u>Determinación de terminaciones de temporizador</u> Es más difícil la determinación del intervalo de expiración, porque el tiempo de tránsito a través de la RED es menos predecible que el tiempo de tránsito a través de un CABLE entre dos ENRUTADORES.</p>
RED	<p><u>Circuitos virtuales versus datagramas en la subred</u> Afecta a la CONGESTIÓN, pues muchos de los ALGORITMOS DE CONTROL DE CONGESTIÓN SÓLO FUNCIONAN CON SUBREDES DE CIRCUITOS VIRTUALES</p> <p><u>Política de encolamiento y servicio de paquetes</u> Los ENRUTADORES deben tener UNA COLA POR CADA LÍNEA DE ENTRADA y UNA O VARIAS COLAS POR CADA LÍNEA DE SALIDA. También se relaciona con el orden en que se procesan los PAQUETES (ejemplo: ROUND ROBIN)</p> <p><u>Política de descartes de paquetes</u> Un PAQUETE se descarta cuando no hay espacio.</p> <p><u>Algoritmo de enrutamiento</u> Puede evitar la CONGESTIÓN si se distribuye el TRÁFICO entre TODAS LAS LÍNEAS.</p> <p><u>Administración de tiempo de vida del paquete</u> Es el tiempo que puede existir un PAQUETE antes de ser DESCARTADO. Si este tiempo es:</p> <ul style="list-style-type: none"> - demasiado grande, los PAQUETES PERDIDOS pueden lo BLOQUEAR LA OPERACIÓN durante un buen rato - demasiado corto, los PAQUETES pueden EXPIRAR antes de llegar a su DESTINO, lo que provoca retransmisiones

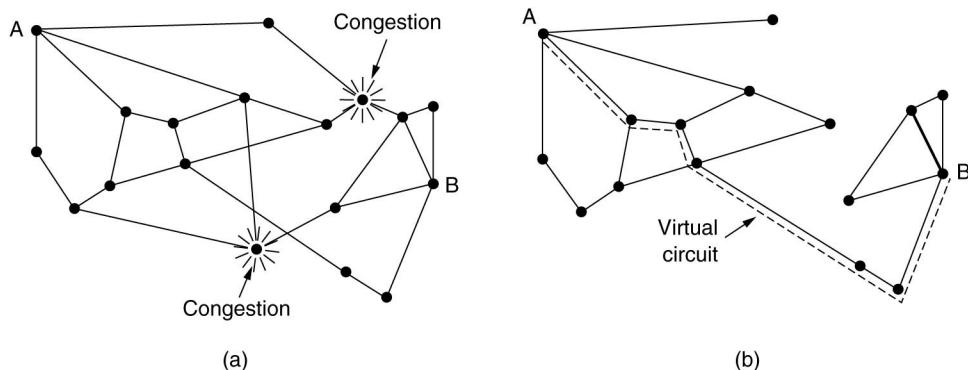
CAPA	POLÍTICAS
ENLACE DE DATOS	<p>Política de retransmisiones Un EMISOR cuyo TEMPORIZADOR expira demasiado pronto y retransmite TODOS LOS PAQUETES usando el PROTOCOLO DE RETROCESO N impondrá una CARGA MAYOR que uno que usa PROTOCOLO DE REPETICIÓN SELECTIVA</p> <p>Política de almacenamiento en caché de PAQUETES fuera de orden Si los RECEPTORES descartan todos los PAQUETES que llegan fuera de orden, luego se tendrán que reenviar causando una CARGA EXTRA</p> <p>Política de confirmación de recepción Si la RECEPCIÓN de cada PAQUETE se confirma de INMEDIATO, los PAQUETES DE CONFIRMACIÓN DE RECEPCIÓN GENERARÁN TRÁFICO EXTRA</p> <p>Política de control de flujo Un esquema de CONTROL DE FLUJO ESTRICTO (VENTANA PEQUEÑA) REDUCE la TASA DE DATOS y permite atacar la CONGESTIÓN</p>

5.3.3 Control de congestión en subredes de circuitos virtuales

Se trata de un MÉTODO para el CONTROL DINÁMICO DE LA CONGESTIÓN para las SUBREDES DE CIRCUITOS VIRTUALES.

Una de las técnicas es el **CONTROL DE ADMISIÓN**: se usa para **evitar que empeoren las CONGESTIONES ya existentes**. Una vez que se ha **detectado la CONGESTIÓN, no se podrán establecer CIRCUITOS VIRTUALES NUEVOS** hasta que el problema haya desaparecido. Los intentos por ESTABLECER CONEXIONES NUEVAS DE CAPA DE RED fallarán.

Un **método alterno** es permitir el **ESTABLECIMIENTO DE NUEVO CIRCUITOS VIRTUALES**, pero **ENRUTANDO CUIDADOSAMENTE los CIRCUITOS NUEVOS por OTRAS RUTAS QUE NO TENGAN PROBLEMAS.**



(a) Subred congestionada

(b) Subred redibujada que elimina la congestión.

También se muestra el circuito virtual de A a B

Otra estrategia es negociar un acuerdo entre el **HOST** y la **SUBRED**, cuando se establezca el **CIRCUITO VIRTUAL**, que asegure la suficiente reserva de recursos:

- **ESPACIO DE TABLAS y BÚFER en los ENRUTADORES**
- **ANCHO DE BANDA EN LAS LÍNEAS**

Para garantizar estos recursos **se negocian los siguientes aspectos**

- **VOLUMEN y FORMA del TRÁFICO**
- **CALIDAD DEL SERVICIO requerida**

La **NEGOCIACIÓN y RESERVACIÓN DE RECURSOS** puede llevarse a cabo durante **todo el tiempo**, o sólo **cuando la SUBRED está CONGESTIONADA**. Realizarlo todo el tiempo provoca un desperdicio de recursos.

5.3.4 Control de congestión en subredes de datagramas

Cada ENRUTADOR puede monitorear el uso de sus LÍNEAS DE SALIDA y de OTROS RECURSOS.

Un ejemplo es asociar a cada LÍNEA DE SALIDA una VARIABLE REAL, u , cuyo valor entre 0.0 y 1.0, refleje su uso reciente. Una buena estimación de u puede requerir la toma de muestras periódicas del uso instantáneo de la LÍNEA, f (0 o 1) y actualizar u periódicamente de acuerdo con

$$U_{\text{nuevo}} = a \cdot U_{\text{anterior}} + (1 - a) \cdot f$$

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

Siempre que el valor de u rebasa un umbral, la LÍNEA DE SALIDA entra en un ESTADO DE ADVERTENCIA y se realiza alguna acción.

El bit de advertencia

La **ARQUITECTURA DECNET antigua** señalaba el **ESTADO DE ADVERTENCIA** activando un bit especial en el **ENCABEZADO DEL PAQUETE**. **FRAME RELAY** también lo hace. Cuando el **PAQUETE** llegaba a su **DESTINO**, la **ENTIDAD TRANSPORTADORA COPIABA EL BIT** en la siguiente **CONFIRMACIÓN DE RECEPCIÓN** que se **REGRESABA AL ORIGEN**. A continuación el **ORIGEN REDUCÍA EL TRÁFICO**. El **ORIGEN MONITOREABA** la fracción de **CONFIRMACIONES DE RECEPCIÓN** que llegaban con el **BIT ACTIVADO** y **AJUSTABA LAS TASAS DE TRANSFERENCIA EN LAS LÍNEAS RESPECTIVAS**.

Paquetes reguladores

En este caso la indicación es directa, mediante un **PAQUETE REGULADOR**.

El **ENRUTADOR** regresa un **PAQUETE REGULADOR** al **HOST DE ORIGEN**, proporcionándole el **DESTINO** encontrado en el **PAQUETE**. El **PAQUETE ORIGINAL** se **ETIQUETA, ACTIVÁNDOLE UN BIT DEL ENCABEZADO**, de manera que **no genere más PAQUETES REGULADORES** y después se **REENVÍA** de la manera usual.

Cuando el **HOST** de **ORIGEN** obtiene el **PAQUETE REGULADOR**, se le pide que **REDUZCA** en un porcentaje X el **TRÁFICO ENVIADO**. Luego el **HOST** debe ignorar otros **PAQUETES REGULADORES** que se refieran a ese **DESTINO** durante un intervalo fijo de tiempo. Transcurrido ese tiempo, el **HOST** escucha más **PAQUETES REGULADORES** durante otro intervalo. Si llegan más **PAQUETES REGULADORES** es porque la **LÍNEA TODAVÍA ESTÁ CONGESTIONADA**. Si no llega ningún **PAQUETE REGULADOR** durante el lapso de escucha, entonces el **HOST** puede incrementar el flujo.

Paquetes reguladores de salto por salto

A **ALTAS VELOCIDADES** o **GRANDES DISTANCIAS**, el **envío de un PAQUETE REGULADOR no funciona bien porque la reacción es muy lenta**.

En estos casos es mejor hacer que el **PAQUETE REGULADOR** ejerza su efecto en cada **SALTO INTERMEDIO** que de por la **SUBRED**.

El **efecto neto** de este esquema de **SALTO POR SALTO** es **proporcionar un ALIVIO RÁPIDO al punto de CONGESTIÓN** a expensas de usar **MÁS BÚFERES ASCENDENTES**. De esta manera **puede cortarse la CONGESTIÓN en la raíz, sin que se pierdan PAQUETES**.

5.3.5 Desprendimiento de carga

Cuando se inunda a los **ENRUTADORES** con **PAQUETES** que no pueden manejar, **simplemente los tiran**.

El **ENRUTADOR** abrumado:

- puede **escoger PAQUETES al azar**
- puede **elegir los PAQUETES a descartar** de acuerdo a las **APLICACIONES QUE SE ESTÁN EJECUTANDO**:
 - En una **TRANSFERENCIA DE ARCHIVOS** son **MÁS IMPORTANTES** los **PAQUETES VIEJOS** que **LOS NUEVOS**, para evitar huecos en los archivos
 - En **MULTIMEDIA** son **MÁS IMPORTANTES** los **PAQUETES NUEVOS** que los **VIEJOS**. También es adecuado implementar una **POLÍTICA DE COOPERACIÓN** entre los **EMISORES**, que es útil en ciertos **ALGORITMOS DE COMPRESIÓN DE VIDEOS**, en los cuáles se transmiten periódicamente una **TRAMA ENTERA** y sus **TRAMAS SUBSIGUIENTES como DIFERENCIAS RESPECTO A LA ÚLTIMA TRAMA COMPLETA**, y por lo cuál es **preferible** que **se descarten** aquellos **PAQUETES** que **corresponden a las DIFERENCIAS**.

Una **POLÍTICA DE DESCARTE** inteligente puede implementarse:

- haciendo que las **APLICACIONES MARQUEN SUS PAQUETES** y estableciendo **CALSES DE PRIORIDADES** para **INDICAR SU IMPORTANCIA**. Se **descartarán** aquellos **PAQUETES con CLASE MÁS BAJA**. Los emisores podrían tener **PERMITIDO ENVIAR PAQUETES DE ALTA PRIORIDAD** bajo **CONDICIONES DE CARGA LIGERA**, pero a medida que **AUMENTE LA CARGA**, los **PAQUETES PODRÍAN DESCARTARSE**
- permitiendo que los **HOSTS excedan** los **límites** especificados en el **acuerdo de negociación** para establecer el **CIRCUITO VIRTUAL** pero sujetos a la **condición** de que el **EXCESO DE TRÁFICO** se **MARQUE** con **PRIORIDAD BAJA**

Detección temprana aleatoria

La idea es descartar PAQUETES antes de que se ocupe todo el espacio de búfer.

El ALGORITMO se conoce **RED(DETECCIÓN TEMPRANA ALEATORIA)(Floyd y Jacobson, 1993)**.

En algunos **PROTOCOLOS DE TRANSPORTE** (entre ellos **TCP**), la reacción ante los **PAQUETES PERDIDOS** es que el **ORIGEN DISMINUYA SU VELOCIDAD DE TRANSMISIÓN**. Este **razonamiento** se debe a que **TCP fue diseñado para REDES CABLEADAS**, y éstas son **MUY CIONFIABLES**, por lo tanto, la **PÉRDIDA DE PAQUETES SE DEBE PRINCIPALMENTE A DESBORDAMIENTOS DE BÚFER** y **NÓ A ERRORES DE TRANSMISIÓN**.

Los ENRUTADORES:

- deben deshacerse de los **PAQUETES** antes de que la situación de **CONGESTIONAMIENTO** sea irremediable.
- para determinar cuando descartar los PAQUETES, mantienen un **PROMEDIO MÓVIL** de **SUS LONGITUDES DE COLA**, entonces **CUANDO LA LONGITUD DE COLA PROMEDIO** en algunas **LÍNEAS SOBREPASA UN UMBRAL**, se determina que la **LÍNEA ESTÁ CONGESTIONADA**
- para **informar al ORIGEN sobre el problema de CONGESTIONAMIENTO** detectado:
 - puede enviarle un **PAQUETE REGULADOR** al ORIGEN, pero esto **aumenta todavía más la CARGA** en la LÍNEA
 - puede **descartar el PAQUETE y NO REPORTAR CONFIRMACIÓN**, entonces el **ORIGEN** notará la falta de **CONFIRMACIONES DE RECEPCIÓN** y reaccionará **REDUCIENDO LA VELOCIDAD DE TRANSMISIÓN DE SUS PAQUETES**. Este **método no funciona en las REDES INALÁMBRICAS**, en las cuáles la **mayoría de las pérdidas se debe al RUIDO en el ENLACE DE RADIO**.

5.3.6 Control de fluctuación

La **DESVIACIÓN ESTÁNDAR** de la **VARIACIÓN EN EL RETARDO** de los **PAQUETES** se conoce como **FLUCTUACIÓN**.

Una **FLUCTUACIÓN ALTA**, por ejemplo cuando **unos PAQUETES tardan** en llegar al **DESTINO 20 mseg y otros 30 mseg**, resultará en una **CALIDAD DESIGUAL** (del **SONIDO** o la **IMAGEN**).

Se debe **escoger un RANGO FACTIBLE** para controlar la **FLUCTUACIÓN** **teniendo en cuenta**:

- el **RETARDO** causado por la **VELOCIDAD DE LA LUZ**
- el **RETARDO MÍNIMO** a través de **los ENRUTADORES**
- un **período adicional corto** para **posibles RETARDOS INEVITABLES**

La **FLUCTUACIÓN** puede **LIMITARSE** calculando el **TIEMPO DE TRÁNSITO ESPERADO PARA CADA SALTO EN LA RUTA**.

Un **PAQUETE** que llega al **ENRUTADOR** se examina para saber que tan **ADELANTADO** o **RETRASADO** está **RESPECTO** a lo **PROGRAMADO**. Esta **información** se **almacena en el PAQUETE** y se **ACTUALIZA** en **CADA SALTO**.

En caso de que:

- el **PAQUETE** esté **ADELANTADO**, el **ENRUTADOR** lo **RETIENE** durante el **tiempo suficiente** para **ajustarse a lo PROGRAMADO**
- el **PAQUETE** esté **RETRASADO**, el **ENRUTADOR** trata de **sacarlo RÁPIDAMENTE**

EL **ALGORITMO** para determinar cuál de varios **PAQUETES** que compiten por una **LÍNEA de SALIDA** debe seguir, **siempre puede escoger el PAQUETE MÁS RETRASADO**, pues de esta manera, los **PAQUETES ADELANTADOS SE FRENAN** y los **RETRASADOS se ACELERAN, DISMINUYENDO la FLUCTUACIÓN**.

En el caso de aplicaciones como el **VIDEO BAJO DEMANDA**, la **FLUCTUACIÓN** puede **ELIMINARSE ALMACENANDO** los **DATOS EN EL BÚFER DEL RECEPTOR** y **DESPUÉS OBTENIÉNDOLOS** de **DICHO BÚFER**, en lugar de utilizar la red en tiempo real. Esto no puede realizarse en aplicaciones que requieren interacción en **TIEMPO REAL**.

5.4 CALIDAD DE SERVICIO

Con el crecimiento de las **REDES DE MULTIMEDIA**, se requieren intentos serios para garantizar la **CALIDAD DEL SERVICIO** a través del **DISEÑO DE REDES** y **PROTOCOLOS**.

5.4.1 Requerimientos

Un **FLUJO** es un **CONJUNTO de PAQUETES** que **van de un ORIGEN a un DESTINO**.

Si la **SUBRED** es:

- **ORIENTADA a la CONEXIÓN**, todos los **PAQUETES** que pertenezcan a un **mismo FLUJO** siguen la **misma RUTA** de su **CIRCUTO VIRTUAL**
- **SIN CONEXIÓN**, basada en **DATAGRAMAS**, los **PAQUETES** pueden seguir diferentes **RUTAS**

La **necesidad de cada FLUJO** puede caracterizarse por:

- **CONFIABILIDAD**
- **RETARDO**
- **FLUCTUACIÓN**
- **ANCHO DE BANDA**

Estos parámetros en conjunto determinan la **QoS (CALIDAD DE SERVICIO)** que el **FLUJO** requiere.

APLICACIÓN	CONFIABILIDAD	RETARDO	FLUCTUACIÓN	ANCHO DE BANDA
Correo electrónico	ALTA	BAJO	BAJA	BAJO
Transferencia de archivos	ALTA	BAJO	BAJA	MEDIO
Acceso Web	ALTA	MEDIO	BAJA	MEDIO
Inicio de sesión remoto	ALTA	MEDIO	MEDIA	BAJO
Audio bajo demanda	BAJA	BAJO	ALTA	MEDIO
Video bajo demanda	BAJA	BAJO	ALTA	ALTO
Telefonía	BAJA	ALTO	ALTA	BAJO
Videoconferencia	BAJA	ALTO	ALTA	ALTO

Las **REDES ATM** clasifican los **FLUJOS** en **CUATRO CATEGORÍAS AMPLIAS** con respecto a sus demandas de QoS:

- 1) **TASA DE BITS CONSTANTE** (ejemplo: telefonía)
- 2) **TASA DE BITS VARIABLE EN TIEMPO REAL** (ejemplo: videoconferencia comprimida)
- 3) **TASA DE BITS VARIABLE NO CONSTANTE** (ejemplo: ver una película a través de **INTERNET**)
- 4) **TASA DE BITS DISPONIBLE** (ejemplo: transferencia de archivos)

Estas **CATEGORÍAS** también son útiles para otros propósitos y otros tipos de **REDES**.

5.4.2 Técnicas para alcanzar buena calidad de servicio

Ninguna técnica proporciona, por sí misma, una QoS eficiente y confiable de una manera óptima. Con frecuencia se combinan múltiples técnicas, como las siguientes.

Sobreaprovisionamiento

Se trata de proporcionar la suficiente **CAPACIDAD de ENRUTADOR, ESPACIO EN BÚFER** y **ANCHO DE BANDA** como para que los **PAQUETES FLUYAN con facilidad**.

Es una solución costosa.

Almacenamiento en búfer

Los **FLUJOS** pueden **ALMACENARSE** en el **BÚFER** en el **LADO DEL RECEPTOR ANTES DE SER ENTREGADOS**. Esto **no afecta la CONFIABILIDAD ni el ANCHO DE BANDA**, aunque **sí INCREMENTA el RETARDO**, pero **ATENÚA la FLUCTUACIÓN**.

Modelado de tráfico

El **ORIGEN** podría emitir los **PAQUETES** con un **espaciado NO UNIFORME** entre ellos, lo cuál **puede causar CONGESTIÓN** en la **SUBRED**. Esta **situación es común** si el **SERVIDOR** está **manejando muchos FLUJOS al mismo tiempo**, y también permite otras acciones como **AVANCE RÁPIDO** y **REBOBINADO**, **AUTENTCACIÓN DE USUARIO**,etc.

El **MODELADO DE TRÁFICO MODERA EL TRÁFICO EN EL SERVIDOR**, en lugar de hacerlo en el **CLIENTE**.

Consiste en **REGULAR LA TASA PROMEDIO (Y LAS RÁFAGAS)** de la **TRANMISIÓN DE LOS DATOS**. Cuando se **ESTABLECE LA CONEXIÓN** el **USUARIO** y la **SUBRED ACUERDAN CIERTO PATRÓN DE TRÁFICO (ACUERDO DE NIVEL DE SERVICIO)**. De esta manera se **REDUCE LA CONGESTIÓN**.

La **SUPERVISACIÓN de UN FLUJO DE TRÁFICO** se conoce como **SUPERVISACIÓN DE TRÁFICO (TRAFFIC POLICING)** llevado a cabo por la **EMPRESA PORTADORA**.

Algoritmo de cubeta con goteo

Hace referencia a la situación de una cubeta con un pequeño agujero en su fondo.

El mismo **CONCEPTO** puede aplicarse a los **PAQUETES** que pasan por los **HOSTS**. Cada **HOST** está conectado a la **SUBRED** mediante una **INTERFAZ DE RED** que contiene una “**cubeta con goteo**”, es decir, una **COLA INTERNA INFINITA**.

Si llega un PAQUETE cuando la **COLA INTERNA ESTÁ LLENA**, éste **SE DESCARTA**.

Este arreglo puede incorporarse en la **INTERFAZ DE HARDWARE** o **SIMULARSE A TRAVÉS DEL SISTEMA OPERATIVO DEL HOST**. Fue propuesto por **Turner (1986)**, se trata de **UN SISTEMA DE ENCOLAMIENTO de un SOLO SERVIDOR con UN TIEMPO de SERVICIO CONSTANTE**.

El **HOST** puede poner en la **RED** un **PAQUETE por PULSO de RELOJ**. Este mecanismo **convierte** un **FLUJO DESIGUAL DE PAQUETES de los PROCESOS DE USUARIO dentro del HOST**, en **UN FLUJO CONTINUO DE PAQUETES HACIA LA RED, MODERANDO las RÁFAGAS y REDUCIENDO** en buena medida las posibilidades de **CONGESTIÓN**.

Cuando se usan **PAQUETES de TAMAÑO VARIABLE**, es **mejor** permitir un **NÚMERO FIJO DE BYTES POR PULSO**, en lugar de un solo PAQUETE. Si el **CONTEO de BYTES RESIDUALES** es **DEMASIADO BAJO**, el siguiente **PAQUETE** deberá **ESPERAR** hasta el **SIGUIENTE PULSO**.

En la **implementación por PAQUETES**:

Si **CUANDO LLEGA UN PAQUETE HAY ESPACIO EN COLA, ÉSTE SE AGREGA A ELLA**
De otro modo, se descarta.

En la **implementación por CONTEO DE BITS**:

- En cada **PULSO** se inicializa un **CONTADOR** en n .
- Si el **PRIMER PAQUETE** de la **COLA** tiene **MENOS BYTES** que el **VALOR ACTUAL DEL CONTADOR, SE TRANSMITE y SE DISMINUYE** el **CONTADOR** en **ESA CANTIDAD DE BYTES**
- Si el **CONTADOR** está **POR DEBAJO** de la **LONGITUD DEL SIGUIENTE PAQUETE** de la **COLA, LA TRANSMISIÓN SE DETIENE HASTA EL SIGUIENTE PULSO.**

Algoritmo de cubeta con tokens

Ahora la **CUBETA CON GOTEÓ** tiene **TOKENS**, generados por un reloj a razón de **un TOKEN por intervalo de tiempo**

Para que se **TRANSMITA** un **PAQUETE**, éste debe **CAPTURAR** y **DESTRUIR** un **TOKEN**.

Este **ALGORITMO DE CUBETA CON TOKENS**:

- ofrece una **forma diferente de modelado de TRÁFICO**
- permite el ahorro, hasta el tamaño máximo de la cubeta, n , significa que **pueden enviarse a la vez ráfagas de hasta n PAQUETES, permitiendo cierta irregularidad en el FLUJO de SALIDA y dando una respuesta más rápida a las ráfagas repentinas**
- **descarta los TOKENS**, es decir, **LA CAPACIDAD DE TRANSMISIÓN, cuando se llena la cubeta pero NUNCA DESCARTA PAQUETES**

Sean:

- **S [seg]** la duración de una ráfaga
- **C [bytes]** la capacidad de la cubeta con tokens
- **p [bytes/seg]** la tasa de llegada de tokens
- **M [bytes/seg]** la tasa máxima de salida

Podemos ver que:

- **una ráfaga de salida contiene un máximo de: $C + pS$ [bytes]**
- **la cantidad de bytes en una ráfaga a velocidad máxima con duración de S segundos es: MS [bytes]**
- **Por lo tanto tenemos: $C + pS = MS$**

Reservación de recursos

Tener la capacidad de regular el **TRÁFICO** ofrecido es un buen inicio para garantizar la **CALIDAD DEL SERVICIO**. Sin embargo usar esta información implica obligar a todos los **PAQUETES** de un **FLUJO** a que sigan la misma **RUTA**, por lo tanto se debe configurar algo similar a un **CIRCUITO VIRTUAL** desde el **ORIGEN** hasta el **DESTINO**, para que todos los **PAQUETES del FLUJO** en cuestión pasen por esta **RUTA**.

Una vez que se tiene una **RUTA** específica para un **FLUJO**, es posible **RESERVAR RECURSOS** a lo largo de esa **RUTA** para asegurar que la capacidad necesaria esté disponible. Se pueden reservar tres tipos de recursos:

1) ANCHO DE BANDA

Significa NO SOBRECARGAR NIGUNA LÍNEA DE SALIDA

2) ESPACIO EN BÚFER

Cuando llega un **PAQUETE** por lo general el hardware mismo lo deposita en la **INTERFAZ DE RED**. A continuación el **SOFTWARE ENRUTADOR** tiene que **COPIARLO** en **UN BÚFER** en **RAM** y colocar en la **COLA** ese **BÚFER** para **TRANSMITIRLO** en una **LÍNEA** de **SALIDA elegida**. Si no hay **BÚFER DISPONIBLE**, el **PAQUETE** se tiene que **DESCARTAR**. Es posible **RESERVAR algunos BÚFERES** para un **FLUJO** específico de manera que no tenga que competir con otros **FLUJOS por ESPACIO EN BÚFER**

3) CICLOS DE CPU

Para procesar un **PAQUETE** se necesita **TIEMPO de CPU del ENRUTADOR**. Un **ENRUTADOR** solo puede procesar cierta cantidad de **PAQUETES POR SEGUNDO**. Hay que **VERIFICAR que la CPU** no esté **SOBRECARGADA**.

Suponiendo:

- que los **PAQUETES** llegan de **MANERA ALEATORIA** con una **TASA PROMEDIO de llegada: n [paquetes/segundo]**
- que el **TIEMPO de CPU** requerido para procesar cada **PAQUETE** también es **ALEATORIO**
- que las **DISTRIBUCIONES DE ARRIBO y SERVICIO**, corresponden a **una DISTRIBUCIÓN DE POISSON**

Es posible demostrar, mediante la **TEORÍA DE ENCOLAMIENTO**, que el **RETARDO PROMEDIO EXPERIMENTADO por un PAQUETE, T**, es:

$$T = \frac{1}{u} * \left(\frac{1}{((1-l)/u)} \right) = \frac{1}{u} * \left(\frac{1}{(1-p)} \right)$$

Donde:

- **p = l/u** es el **USO de CPU**
- **1/u** es el **TIEMPO DE SERVICIO SI NO HUBIERA COMPETENCIA**
- El **SEGUNDO FACTOR** es la **REDUCCIÓN DE VELOCIDAD DEBIDO a la COMPETENCIA CON OTROS FLUJOS**

Control de admisión

Hasta ahora el **TRÁFICO ENTRANTE** de algún **FLUJO** está **bien MODELADO** y puede **seguir una SOLA RUTA**. Cuando un **FLUJO** así se ofrece a un **ENRUTADOR**, éste tiene que **DECIDIR**, basándose en su **CAPACIDAD** y sus **COMPROMISOS YA ADQUIRIDOS CON OTROS FLUJOS, si lo ADMITE o lo RECHAZA**. Esta decisión es bastante complicada para el **ENRUTADOR**, pues éste debe considerar que **ALGUNAS APLICACIONES**

- podrían saber sus **REQUERIMIENTOS** de **ANCHO DE BANDA**, pero **poco acerca de BÚFERES o CICLOS DE CPU**, por lo que se **necesita** una **forma diferente de DESCRIBIR LOS FLUJOS**
- son mucho **más tolerantes** con el **incumplimiento ocasional de plazos**
- podrían estar **dispuestas a negociar los parámetros del FLUJO**

Como muchas partes están involucradas en la **NEGOCIACIÓN DEL FLUJO** (EMISOR, RECEPTOR, y **TODOS LOS ENRUTADORES** a lo largo de la RUTA) los **FLUJOS** se deben **DESCRIBIR** de manera **PRECISA** en términos de **PARÁMETROS ESPECÍFICOS** que se puedan negociar.

Un **CONJUNTO** de **TALES PARÁMETROS** se conoce como **ESPECIFICACIÓN DE FLUJO**.

Por lo general el **EMISOR** (ejemplo: **SERVIDOR DE VIDEO**) **PRODUCE UNA ESPECIFICACIÓN DE FLUJO** que propone los parámetros que le gustaría utilizar. Conforme la **ESPECIFICACIÓN** se **PROPAGUE** por la **RUTA**, cada **ENRUTADOR** la examina y modifica los parámetros según sea necesario. **Sólo está permitido REDUCIR EL FLUJO, NO INCREMENTARLO.**

Un **ejemplo de una ESPECIFICACIÓN DE FLUJO**, basado en los **RFC 2210** y **2211**, se muestra a continuación usando los siguientes **PARÁMETROS**:

- **TASA DE LA CUBETA CON TOKENS [Bytes/seg]**
Es la cantidad de bytes por segundo que se colocan en la cubeta y también la tasa máxima que el EMISOR puede TRANSMITIR PROMEDIADA con respecto a un intervalo de tiempo largo
- **Tamaño de la CUBETA [Bytes]**
- **Tasa pico de datos [Bytes/seg]**
Es la tasa máxima de transmisiones tolerada, incluso durante un intervalo de tiempo breve. El EMISOR nunca debe sobrepasar esta tasa.
- **Tamaño mínimo de paquete [Bytes]**
Es importante porque procesar cada PAQUETE toma un tiempo fijo.

- **Tamaño máximo de paquete [Bytes]**

Es importante debido a las limitaciones internas de la RED que no deben sobrepasarse

El tema es **cómo convierte** un **ENRUTADOR** una **ESPECIFICACIÓN DE FLUJO** en un **CONJUNTO DE REVERSACIONES DE RECURSOS ESPECÍFICOS**, pues **esta CONVERSIÓN NO ESTÁ ESTANDARIZADA** y **DEPENDE DE CADA IMPLEMENTACIÓN PROPIETARIA**.

Enrutamiento proporcional

La mayoría de los **ALGORITMOS DE ENRUTAMIENTO** tratan de encontrar la **MEJOR RUTA PARA CADA DESTINO** y **ENVÍAN A TRAVÉS DE ELLAS TODO EL TRÁFICO**.

Un **método diferente** para **proporcionar una CALIDAD DE SERVICIO MÁS ALTA** es **DIVIDIR el TRÁFICO** a través de **MÚLTIPLES RUTAS** utilizando la **información disponible localmente** en cada **ROUTER**.

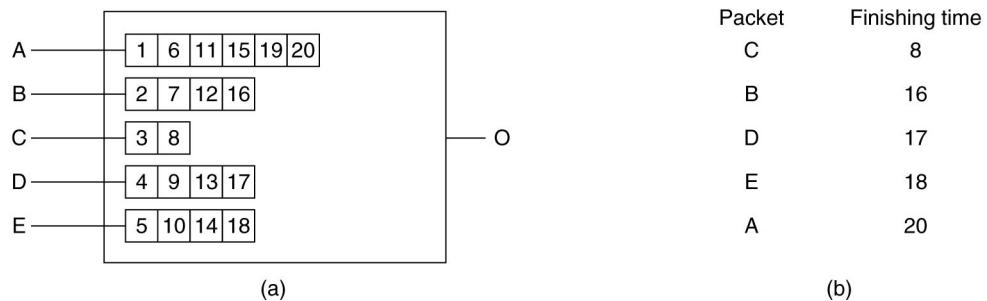
Calendarización de paquetes

Un **ENRUTADOR** maneja **MÚLTIPLES FLUJOS** y está latente el peligro de que **un FLUJO acapare mucha de su capacidad y limite a los otros FLUJOS**, reduciendo la **CALIDAD DE SERVICIO** para esos **FLUJOS**.

El **ALGORITMO DE ENCOLAMIENTO JUSTO (FAIR QUEUEING, Nagle 1987)** consiste en que los **ENRUTADORES** tengan **COLAS SEPARADAS PARA CADA LÍNEA DE SALIDA, UNA POR FLUJO**. Entonces **cuando una LÍNEA QUEDA INACTIVA**, el **ROUTER** explora las diferentes **colas de manera circular (ROUND ROBIN)** y toma el primer **PAQUETE** de la siguiente **COLA**. Con **N HOST** compitiendo por una **LÍNEA DE SALIDA**, cada **HOST** obtiene la oportunidad de **ENVIAR uno de N PAQUETES**.

Al principio el **problema es** que **proporciona más ANCHO DE BANDA a los HOST que utilizan PAQUETES MÁS GRANDES**.

Una **mejora**, sugerida por **Demers y cols. (1990)** es que **se realice la exploración circular de tal manera que simule una exploración circular byte por byte**. Va explorando las **COLAS** de manera repetida, byte por byte, hasta que **encuentra el fin de cada paquete**. A continuación, **los PAQUETES se ordenan conforme a su tiempo de terminación y se envían en ese orden**.



- (a) Un ENRUTADOR con cinco PAQUETES ENCOLADOS en la LÍNEA O
 (b) Tiempos de terminación de los cinco PAQUETES

Un problema con este algoritmo es que asigna la misma prioridad para todos los **HOSTS**. A fin de que se pueda dar mayor prioridad (y **ANCHO DE BANDA**) a uno de los **SERVIDORES** (como un **SERVIDOR DE VIDEO**) que a otros (como un **SERVIDOR DE ARCHIVOS**) se utilizan dos o más **BYTES POR PULSOS**. Este algoritmo se conoce como **ENCOLAMIENTO JUSTO PONDERADO (WEIGHTED FAIR QUEUEING)** y se utiliza ampliamente.

El reenvío real de PAQUETES a través de un ENRUTADOR o CONMUTADOR SE ESTÁ REALIZANDO ACTUALMENTE CADA VEZ MÁS EN HARDWARE.

5.4.3 Servicios integrados

Entre **1995** y **1997**, la **IETF**(Internet Engineering Task Force) se dedicó a diseñar una **ARQUITECTURA para la MULTIMEDIA DE FLUJOS CONTINUOS**, que quedó plasmado en casi dos docenas de **RFCs, 2205-2210**. Este trabajo se conoce como **ALGORITMOS BASADOS EN FLUJO** o **SERVICIOS INTEGRADOS**.

Sirve tanto para aplicaciones:

- de **UNIDIFUSIÓN**, por ejemplo: **un solo usuario difundiendo un clip de video**
- de **MULTIDIFUSIÓN**, por ejemplo: **UNA COLECCIÓN DE ESTACIONES DE TELEVISIÓN DIGITAL DIFUNDIENDO** sus programas como **FLUJOS de PAQUETES IP** a **MUCHOS RECEPTORES DE DIFERENTES UBICACIONES**
 En muchas de estas aplicaciones **los grupos pueden cambiar su membresía en forma dinámica**, por lo que el **método de hacer que los EMISORES reserven ANCHO DE BANDA por adelantado** no funciona bien, pues **requeriría que cada EMISOR rastreara TODAS las ENTRADAS y SALIDAS** de su audiencia.

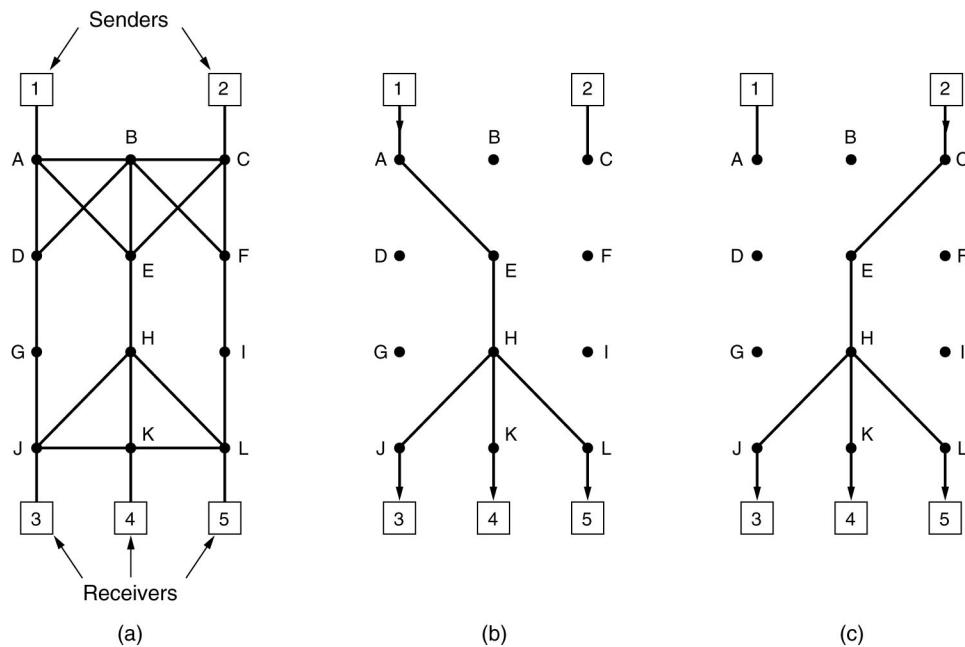
RSVP – Protocolo de reservación de recursos

Este protocolo se describe en el **RFC 2205 y en otros**.

Se utiliza **para marcar las reservas**, no se usa para el envío de datos.
Entre otras cosas **RSVP**:

- permite que **VARIOS EMISORES** transmitan a **MÚLTIPLES GRUPOS** de **RECEPTORES**
- permite que **RECEPTORES INDIVIDUALES CAMBIEN DE CANAL LIBREMENTE**
- optimiza el uso del **ANCHO DE BANDA**
- elimina la **CONGESTIÓN**

En su forma más sencilla, el **PROTOCOLO usa ENRUTAMIENTO de MULTIDIFUSIÓN CON ÁRBOLES de EXPANSIÓN. A CADA GRUPO se le asigna un GRUPO de DIRECCIONES**. Cuando un **EMISOR ENVÍA** a un **GRUPO**, pone la **DIRECCIÓN del GRUPO** en sus **PAQUETES**. Mediante el **ARBOL DE EXPANSIÓN cubre a TODOS LOS MIEMBROS del GRUPO**. De esta manera envía una **MULTIDIFUSIÓN con un POCO DE INFORMACIÓN EXTRA** en forma periódica para informar a los **ENRUTADORES** a lo largo del **ARBOL** que mantengan **CIERTAS ESTRUCTURAS DE DATOS** en sus **MEMORIAS**.

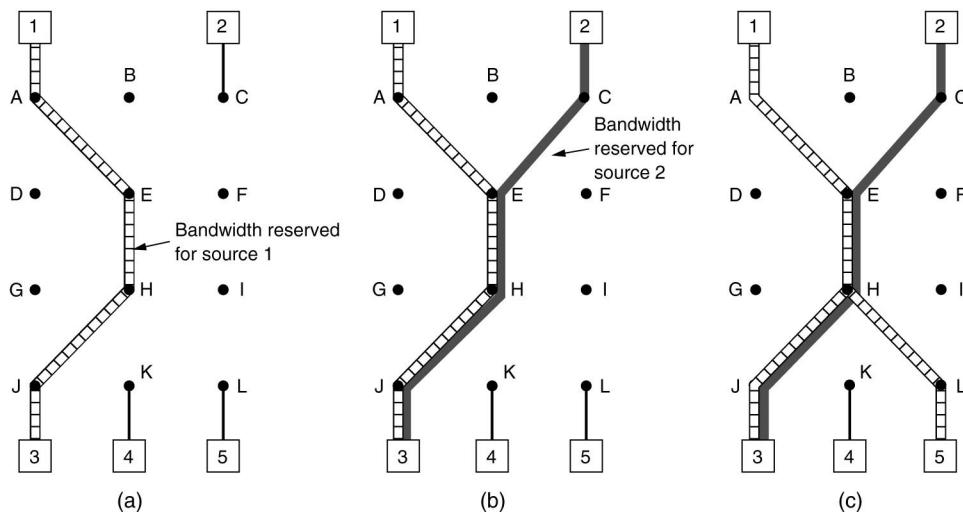


(a) Los HOSTS **1** y **2** son **EMISORES MULTIDIFUSIÓN (SERVIDORES)**
Los HOSTS **3, 4** y **5** son **RECEPTORES MULTIDIFUSIÓN (CLIENTES)**

(b) **ÁRBOL DE EXPANSIÓN DE MULTIDIFUSIÓN** del HOST **1**
(c) **ÁRBOL DE EXPANSIÓN DE MULTIDIFUSIÓN** del HOST **2**

En general los **GRUPOS** pueden traslaparse.

Para obtener **MEJOR RECEPCIÓN** y **ELIMINAR CONGESTIÓN**, los **RECEPTORES** de un **GRUPO** envían un mensaje por el **ÁRBOL DE EXPANSIÓN** al **EMISOR** usando el **ALGORITMO DE REENVÍO POR RUTA INVERTIDA**. Los **ENRUTADORES** van notando la **RESERVACIÓN** y **APARTANDO el ANCHO DE BANDA NECESARIO**, si no pueden hacerlo informan de una **FALLA**. Al llegar el mensaje nuevamente al **ORIGEN**, ya se han **RESERVADO** los **RECURSOS DESDE EL EMISOR HASTA EL RECEPTOR**.



- El HOST 3 SOLICITA UN CANAL al HOST 1
- El HOST 3 SOLICITA UN SEGUNDO CANAL al HOST 2
- El HOST 5 SOLICITA UN CANAL al HOST 1

Dos **RECEPTORES** pueden **COMPARTIR UNA RUTA** si ambos están de acuerdo en no cambiar los **ORÍGENES** posteriormente.

Una vez que un **RECEPTOR** ha **RESERVADO ANCHO DE BANDA**, puede **COMUTARSE a OTRO ORIGEN**, ofreciendo un buen **DINAMISMO**.

5.4.4 Servicios diferenciados

Los **ALGORITMOS** basados en **FLUJO** tienen como desventaja el requisito de una configuración avanzada para establecer cada **FLUJO**. Además mantienen el estado por **FLUJO** interno de los **ENRUTADORES**, haciéndolos **VULNERABLES a la CAÍDA DE LOS ROUTERS** y los cambios requeridos al **CÓDIGO DE ENRUTADOR** son **SUSTANCIALES** e involucran **intercambios complejos de ENRUTADOR a ENRUTADOR**.

Por estos motivos el **IETF** ha diseñado un método más simple para la **CALIDAD DE SERVICIO BASADA EN CLASE**. La **ARQUITECTURA** se describe en los **RFCs 2474, 2475** entre otros. Puede **implementarse** de manera local en cada **ENRUTADOR SIN UNA CONFIGURACIÓN AVANZADA NI NECESIDAD DE INVOLUCRAR A TODA LA RUTA**.

Un **CONJUNTO** de **ENRUTADORES** forman un **DOMINIO ADMINISTRATIVO** (ejemplo: **ISP** o **COMPAÑÍA TELEFÓNICA**) que puede ofrecer los **SERVICIOS DIFERENCIADOS**, “**DS**”, en los que **DEFINE sus propias REGLAS DE REENVÍO**. Entonces **sus CLIENTES** suscriptores, **tendrán en sus PAQUETES** un **CAMPO TIPO DE SERVICIO** que **especifica la CLASE DE SERVICIO** a la que pertenece (ejemplo: PREMIUM).

Este esquema **NO REQUIERE** de una **CONFIGURACIÓN AVANZADA, NI RESERVA DE RECURSOS, NI NEGOCIACIÓN EXTREMO A EXTREMO**.

Reenvío expedito o acelerado

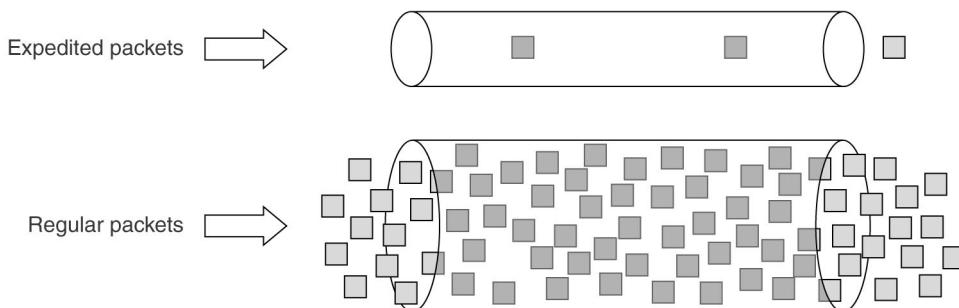
Cada **OPERADOR DE SUBRED** elige **sus propias CLASES DE SERVICIOS**, pero como los **PAQUETES** con frecuencia **SE REENVÍAN ENTRE SUBLDES DE OPERADORES DIFERENTES**, la **IETF** está trabajando para definir las **CLASES DE SERVICIOS INDEPENDIENTES DE LA SUBRED**.

Hay **DOS CLASES DE SERVICIOS DISPONIBLES: REGULAR y EXPEDITA**.

El **REENVÍO EXPEDITO** se describe en el **RFC 3246**.

Se espera que la **MAYOR PARTE** del **TRÁFICO sea REGULAR** y una **PEQUEÑA FRACCIÓN** de los **PAQUETES EXPEDITA**.

Los **PAQUETES EXPEDITOS** deben tener la **CAPACIDAD de TRANSITAR la SUBRED COMO SI NO HUBIERAN OTROS PAQUETES**.



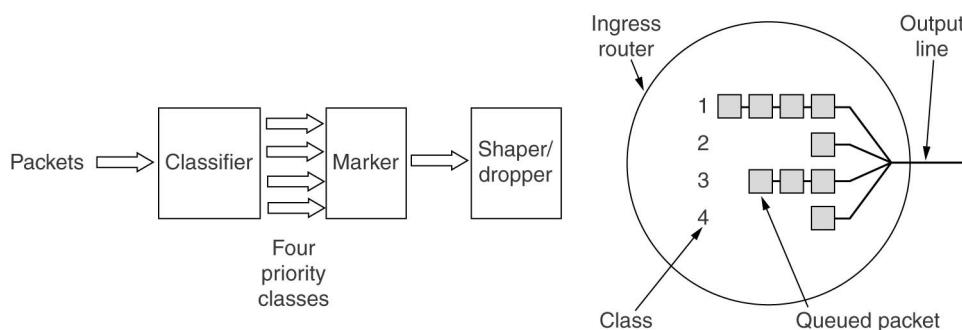
Estos dos conductos lógicos representan una forma de reservar ANCHO DE BANDA, no son dos líneas separadas

Una **forma de implementarlo** es disponiendo en los **ENRUTADORES DOS COLAS DE SALIDA** por **CADA LÍNEA DE SALIDA**, una para los **PAQUETES EXPEDITOS** y la otra para los **REGULARES**.

Cuando **llega un PAQUETE** al **ENRUTADOR**, se **coloca en la COLA correspondiente**. Los **PAQUETES** se **PROCESAN** utilizando algo parecido al **ENCOLAMIENTO JUSTO PONDERADO**.

Reenvío asegurado

Es un esquema un poco más elaborado para el manejo de las CLASES DE SERVICIOS. Se describe en el RFC 2597 y especifica la existencia de CUATRO CLASES DE PRIORIDADES, cada una de las cuales tendrá sus propios recursos. Además define TRES PROBABILIDADES DE DESCARTE PARA PAQUETES QUE ESTÁN EN CONGESTIÓN: ALTA, MEDIA, BAJA. Ambos factores juntos definen un total de 12 CLASES DE SERVICIOS.



Una forma en que los PAQUETES pueden ser PROCESADOS bajo REENVÍO ASEGURADO es la siguiente:

- PASO 1:
Se CLASIFICAN los PAQUETES según su CLASE DE PRIORIDAD.
Se puede hacer en el HOST EMISOR, donde se dispone de más información acerca de cuáles PAQUETES pertenecen a qué FLUJOS, o en el ENRUTADOR.
- PASO 2:
Se marcan los PAQUETES de acuerdo con su CLASE.
En el ENCABEZADO IP, se dispone de un CAMPO: TIPO DE SERVICIO, de 8 bits, de los cuales se usan 6 de estos bits para marcar a qué CLASE pertenecen
- PASO 3:
Se pasan los PAQUETES por un FILTRO MODELADOR/ELIMINADOR que podría RETARDAR o DESCARTAR algunos de ellos, para dar forma aceptable a los CUATRO FLUJOS, por ejemplo mediante CUBETA CON GOTEOS o TOKENS

También son posibles esquemas elaborados que involucren la medición y la retroalimentación.

Estos tres pasos se realizan en el HOSTS EMISOR, por lo que el FLUJO DE SALIDA ahora se introduce en el ENRUTADOR DE INGRESO. Los pasos pueden ser realizados por SOFTWARE ESPECIAL DE CONECTIVIDAD DE REDES o por el mismo SISTEMA OPERATIVO a fin de no tener que cambiar las aplicaciones existentes.

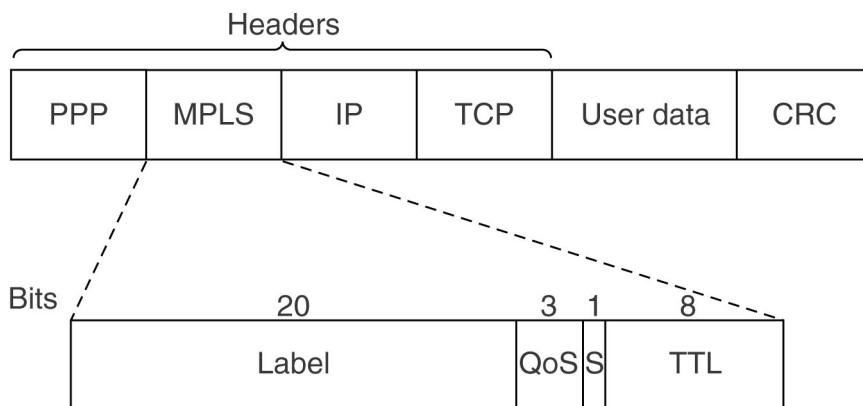
5.4.5 Comutación de etiquetas y MPLS

Varios fabricantes de **ENRUTADORES** han desarrollado **métodos de reenvío** agregando una **ETIQUETA** en el **FRENTE DE CADA PAQUETE**, y **enrutando en base a esta ETIQUETA**, sin usar la **DIRECCIÓN DE DESTINO**. La **ETIQUETA** se **usa como ÍNDICE en una TABLA DE ENRUTAMIENTO**, de esta manera se **disminuye el tiempo** requerido para **procesar cada PAQUETE** con una **simple búsqueda del ÍNDICE** en la **TABLA** y cualesquier **RECURSOS** pueden **RESERVARSE A LO LARGO DE LA RUTA**.

Esta manera de **ETIQUETAR** los **FLUJOS** es muy similar a las **SUBREDDES DE CIRCUITOS VIRTUALES**, como **X.25, ATM, FRAME RELAY** y otras.

El **IETF** comenzó a estandarizar esta idea bajo la denominación **MPLS**, **“CONMUTACIÓN DE ETIQUETAS MULTIPROTOCOLO”**, en el **RFC 3031**. El primer problema fue dónde colocar la **ETIQUETA**, pues los **PAQUETES IP NO FUERON DISEÑADOS PARA CIRCUITOS VIRTUALES** y en el **ENCABEZADO IP no había CAMPO disponible** para los **números de tales circuitos**. Por lo tanto se tuvo que **AGREGAR UN NUEVO ENCABEZADO EN FRENTE DEL ENCABEZADO IP**.

En una **LÍNEA ENRUTADOR-ENRUTADOR** que utiliza **PPP** como **PROTOCOLO DE CAPA DE ENLACE**, **EL FORMATO DE LA TRAMA, incluyendo los ENCABEZADOS PPP, MPLS, IP y TCP** luce como se muestra a continuación:



El **ENCABEZADO MPLS** tiene los **CUATRO** siguientes **CAMPOS**:

- **ETIQUETA:**
El más importante, pues contiene el **ÍNDICE**
- **QoS (bits experimentales):**
Indica la CLASE DE SERVICIO
- **S:**
Se relaciona con colocar en **UNA PILA MÚLTIPLES ETIQUETAS EN REDES JERÁRQUICAS** (de varios niveles)
Si el valor del bit **S** es:
 - **1**, indica que es la **última ETIQUETA añadida al PAQUETE IP**
 - **0**, indica que **hay más ETIQUETAS añadidas al PAQUETE IP**

- **TTL (Time to Live):**

Evita el **CICLO INFINITO** en caso de **inestabilidad** en el **ENRUTAMIENTO**, pues se **DECREMENTA EN CADA ENRUTADOR** y **AL LLEGAR A CERO EL PAQUETE SE DESCARTA**

Pero **MPLS** es **independiente** de la **CAPA DE RED** o **DE ENLACE DE DATOS** y **es posible construir CONMUTADORES MPLS que pueden REENVIAR tanto PAQUETES IP como CELDAS ATM**, en función de lo que aparezca.

Cuando un **PAQUETE**, o **CELDA**, “**MEJORADO con MPLS**” llega a un **ENRUTADOR CON CAPACIDAD MPLS**, la **ETIQUETA** se usa como **ÍNDICE** en una **TABLA** para **DETERMINAR** la **LÍNEA DE SALIDA** y la **NUEVA ETIQUETA** a utilizar en el próximo **ENRUTADOR**, por lo tanto la **ETIQUETA NO SE MANTIENE A LO LARGO DE LA RUTA**.

Es posible que cada **FLUJO** tenga su propio conjunto de etiquetas a través de la **SUBRED**, sin embargo **es más común** que los **ENRUTADORES AGRUPEN MÚLTIPLES FLUJOS** que **terminan en un ROUTER o una LAN** y **utilicen UNA MISMA ETIQUETA PARA TODOS ELLOS**, en este caso **se dice** que los **FLUJOS pertenecen a la misma FEC (CLASE DE EQUIVALENCIA DE REENVÍO)**.

Existen **DOS FORMAS DE CREAR LAS ENTRADAS EN LA TABLA DE ENRUTAMIENTO:**

- **MÉTODO ORIENTADO A DATOS:**

Cuando el **PAQUETE** llega, el **primer ENRUTADOR** que lo encuentra **contacta al siguiente** en sentido descendente del **FLUJO** y **le pide que genere una ETIQUETA para el FLUJO**. Se **aplica recursivamente**, evitando la creación de **CICLOS** mediante una técnica llamada **SUBPROCESOS CON COLOR (COLORED THREADS)**.
Se usa principalmente en redes en que el TRANSPORTE SUBYACENTE ES ATM (mayor parte del sistema telefónico)

- **MÉTODO DIRIGIDO POR CONTROL:**

Tiene algunas variantes. Cuando se **inicia un ENRUTADOR**, verifica para cuáles **RUTAS** es el **último SALTO**. Crea una o más **FEC** para ellas, asigna una **ETIQUETA** para cada una de ellas y pasa las **ETIQUETAS a sus VECINOS**. Estos a su vez, introducen las **ETIQUETAS en sus TABLAS DE REENVÍO** y **envían nuevas ETIQUETAS a sus VECINOS**, hasta que todos los **ENRUTADORES** han adquirido la **RUTA**. También **es posible reservar recursos conforme se construye la RUTA** para garantizar una **CALIDAD DE SERVICIO** apropiada

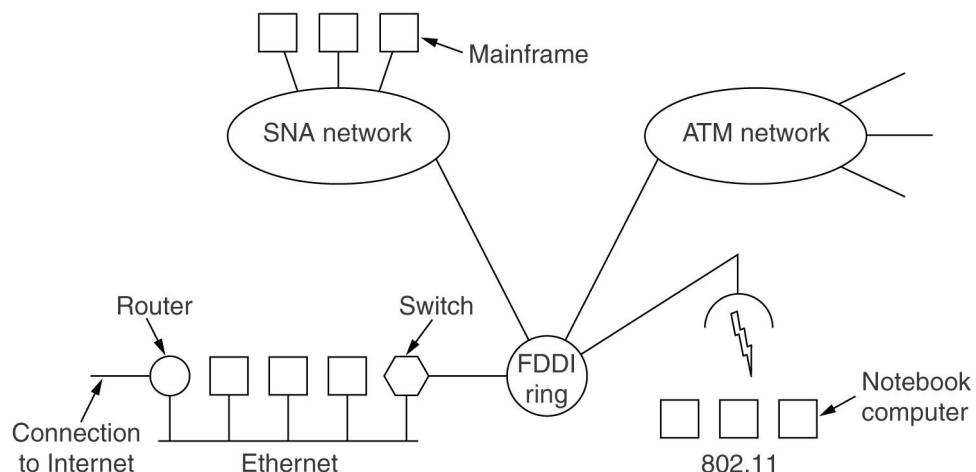
5.5 INTERCONECTIVIDAD

Existen muchas **REDES DIFERENTES, LANs, MANs y WANs**. En cada **CAPPA** hay numerosos **PROTOCOLOS DISTINTOS** de uso muy difundido. Es por esto que **surgen problemas** cuando **dos o más REDES SE UNEN FORMANDO UNA INTERRED**.

Creemos que siempre habrá una variedad de REDES y PROTOCOLOS DIFERENTES pues:

- hay muchas instalaciones **UNIX** ejecutando **TCP/IP**
- muchos **negocios grandes** aún tienen **MAINFRAMES** ejecutando **SNA de IBM**
- muchas **COMPAÑÍAS TELEFÓNICAS** operan **ATM**
- algunas **LANs de computadoras personales** aún usan **NOVEL NCP/IPX o APPLETALK**
- las recientes **REDES INALÁMBRICAS** en desarrollo y con una variedad de **PROTOCOLOS**
- a medida que las **computadoras y las redes se abaratan**, la toma de decisiones se desplaza hacia abajo
- las **tecnologías que usan las distintas redes son radicalmente diferentes** y a medida que hayan **nuevos avances en HARDWARE**, necesariamente deberá **implementarse el SOFTWARE adaptado** para tales novedades

Se muestra a continuación un **ejemplo de cómo se pueden interconectar redes diferentes**:



El propósito de interconectar todas estas redes es permitir que los usuarios de cualquiera de ellas :

- se comuniquen con los demás
- accedan a los datos de los demás

5.5.1 Cómo difieren las redes

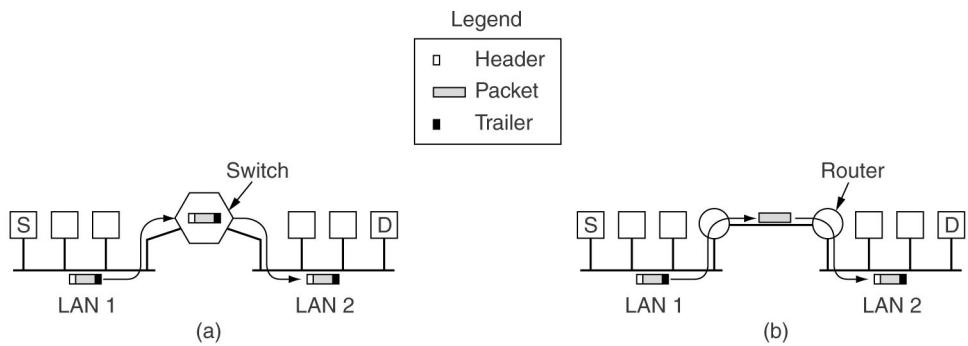
Las redes pueden diferir de muchas maneras como se muestra en la siguiente tabla, a nivel de **CAPA DE RED**:

ASPECTO	ALGUNAS POSIBILIDADES
Servicio ofrecido	Sin conexiones, orientado a conexiones
Protocolos	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc
Direccionamiento	Plano (802) o jerárquico (IP)
Multidifusión	Presente o ausente (también difusión)
Tamaño de paquete	Cada red tiene su propio máximo
Calidad de servicio	Puede estar presente o ausente; muchos tipos diferentes
Manejo de errores	Entrega confiable, ordenada y desordenada
Control de flujo	Ventana corrediza, control de tasa, otros o ninguno
Control de congestión	Cubeta con goteo, paquetes reguladores, etc
Seguridad	Reglas de confidencialidad internacionales, encriptación, etc
Parámetros	Diferentes terminaciones de temporizador, especificaciones de flujo, etc
Contabilidad	Por tiempo de conexión, por paquete, por byte, o sin ella

5.5.2 Conexión de redes

Las redes pueden interconectarse mediante diversos dispositivos como vemos en la siguiente tabla:

CAPA	DISPOSITIVOS	CARACTERÍSTICAS
APLICACIÓN	PUERTA DE ENLACE DE APLICACIÓN (APPLICATION GATEWAY)	Traducen SEMÁNTICAS de MENSAJE (ejemplo: las puertas de enlace entre el correo electrónico de Internet, RFC 822, y el correo electrónico X.400, que deben analizar los mensajes de correo electrónico y cambiar varios campos de encabezado)
TRANSPORTE	PUERTA DE ENLACE DE TRANSPORTE (TRANSPORT GATEWAY)	Pueden interactuar entre dos conexiones de transporte (ejemplo: una PUERTA DE ENLACE DE TRANSPORTE podría permitir que los PAQUETES fluyeran ENTRE una RED TCP y una SNA)
RED	ENRUTADORES (ROUTERS)	Pueden conectar dos redes. Si las redes tienen CAPA DE RED DIFERENTE pueden tener la CAPACIDAD DE TRADUCIR ENTRE LOS FORMATOS DIFERENTES DE PAQUETES Pueden ser ENRUTADORES MULTIPROTOCOLO
ENLACE DE DATOS	CONMUTADORES (SWITCHES) o PUENTES (BRIDGES)	Aceptan TRAMAS Examinan las DIRECCIONES MAC Reenvían las TRAMAS a una RED DIFERENTE MIENTRAS REALIZAN UNA TRADUCCIÓN MENOR DE PROTOCOLOS EN EL PROCESO (ejemplo: Ethernet a FDDI o a 802.11)
FÍSICA	REPETIDORES o CONCENTRADORES	Mueven bits de una red a otra idéntica En su gran mayoría son dispositivos analógicos que simplemente regeneran señales eléctricas (no entienden nada sobre protocolos digitales)



Header: ENCABEZADO

Packet: PAQUETE

Trailer: TERMINADOR

(a) Conectividad en CAPA DE ENLACE DE DATOS: dos Ethernets conectadas mediante un CONMUTADORES

(b) Conectividad en CAPA DE RED: dos Ethernets conectadas mediante ROUTERS

Veamos primero cómo funciona la conectividad en la **CAPA DE ENLACE DE DATOS**:

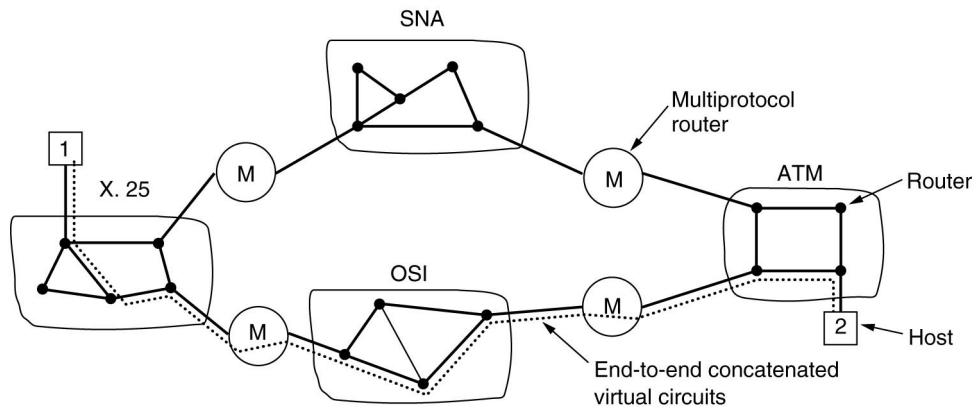
- El **HOST S** desea enviar un **PAQUETE** al **HOST D**
- El **HOST S** **ENCAPSULA** el **PAQUETE** (de la **CAPA DE RED**) en una **TRAMA** (de la **CAPA DE ENLACE DE DATOS**) y **lo envía al destino**
- La **TRAMA** llega al **CONMUTADOR**, éste examina **LA DIRECCIÓN MAC** de dicha **TRAMA** y determina que debe enviarlo hacia la **LAN 2**, entonces el **CONMUTADOR ELIMINA LA TRAMA** de la **LAN 1** y la **COLOCA** en la **LAN 2**

Ahora el caso en que las dos Ethernets están interconectadas mediante un ENRUTADOR dispuesta en cada LAN y conectados mediante una LÍNEA ALQUILADA DE PUNTO A PUNTO DE MILES DE KILOMETROS:

- El **HOST S** desea enviar un **PAQUETE** al **HOST D**
- El **HOST S** **ENCAPSULA** el **PAQUETE** (de la **CAPA DE RED**) en una **TRAMA** (de la **CAPA DE ENLACE DE DATOS**) y **lo envía al destino**
- El **ENRUTADOR de la LAN 1**:
 - Recoge la **TRAMA** y de su **CAMPO de DATOS**, extrae el **PAQUETE**
 - Examina **LA DIRECCIÓN DEL PAQUETE** (ejemplo: **DIRECCIÓN IP**), la **BUSCA** en su **TABLA DE ENRUTAMIENTO** y **DETERMINA** que **debe enviar** el **PAQUETE** hacia el **ENRUTADOR REMOTO** que está en la **LAN 2**
 - **ENCAPSULA** el **PAQUETE** en un **TIPO DIFERENTE DE TRAMA** que **depende** del **PROTOCOLO de LÍNEA usado** y lo envía a través de la **LÍNEA**
- El **ENRUTADOR de la LAN 2**:
 - Recoge de la **LÍNEA** el **TIPO DIFERENTE DE TRAMA**
 - Extrae del **CAMPO DE DATOS** del **TIPO DIFERENTE DE TRAMA** el **PAQUETE**
 - Deposita el **PAQUETE** en el **CAMPO DE DATOS** de una **TRAMA ETHERNET**
 - Deposita la **TRAMA ETHERNET** en la **LAN 2**

5.5.3 Circuitos virtuales concatenados

El modelo de **CIRCUITOS VIRTUALES CONCATENADOS** luce como en el siguiente esquema:



El establecimiento de una conexión con un **HOST** de una red distante es **similar a las conexiones normales**.

La **SUBRED** nota que el **DESTINO es REMOTO** y **CONSTRUYE un CIRCUITO VIRTUAL hacia el ENRUTADOR MÁS CERCANO a la RED DE DESTINO**. Continúa construyendo el **CIRCUITO VIRTUAL desde ese ENRUTADOR hacia la PUERTA DE ENLACE EXTERNA (ENRUTADOR MULTIPROTOCOLO)**, que **REGISTRA la existencia del CIRCUITO VIRTUAL en sus TABLAS y PROCEDE a CONSTRUIR otro CIRCUITO VIRTUAL a un ENRUTADOR en la SIGUIENTE SUBRED**.

El proceso se repite hasta llegar finalmente al **HOST DESTINO**.

Cuando comienzan a fluir **PAQUETES de DATOS** por la RUTA, cada **PUERTA DE ENLACE INTERMEDIA retransmite los PAQUETES DE ENTRADA y HACE LAS CONVERSIONES NECESARIAS ENTRE LOS FORMATOS DE PAQUETE y los NÚMEROS DE CIRCUITO VIRTUAL**, según sea necesario.

La red **NUNCA REORDENA LOS PAQUETES DE UN FLUJO**.

EL **ESQUEMA FUNCIONA MEJOR CUANDO TODAS LAS REDES INTERMEDIAS POSEEN LAS MISMAS PROPIEDADES**, que en general es muy distinto en realidad.

Las **ventajas** son las mismas que tienen los **CIRCUITOS VIRTUALES** en una sola subred:

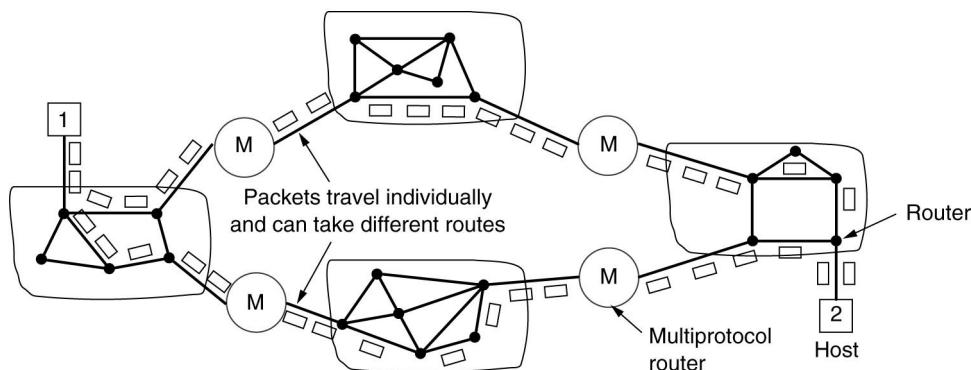
- pueden **RESERVARSE BÚFERES** por adelantado
- puede garantizarse la **SECUENCIA**
- pueden usarse **ENCABEZADOS más cortos**
- pueden evitarse los problemas causados por los **PAQUETES DUPLICADOS RETRASADOS**

También presenta **las mismas desventajas** que **los CIRCUITOS VIRTUALES en una sola SUBRED**:

- requiere **ESPACIO** en las **TABLAS de los ENRUTADORES**
- falta de **RUTAS ALTERNATIVAS** para evitar **CONGESTIONES**
- su **implementación es muy difícil si una de las subredes involucradas es una SUBRED NO CONFIABLE DE DATAGRAMAS**

5.5.4 Interconectividad no orientada a la conexión

Vemos como luce el **MODELO DE INTERCONECTIVIDAD MEDIANTE DATAGRAMAS**:



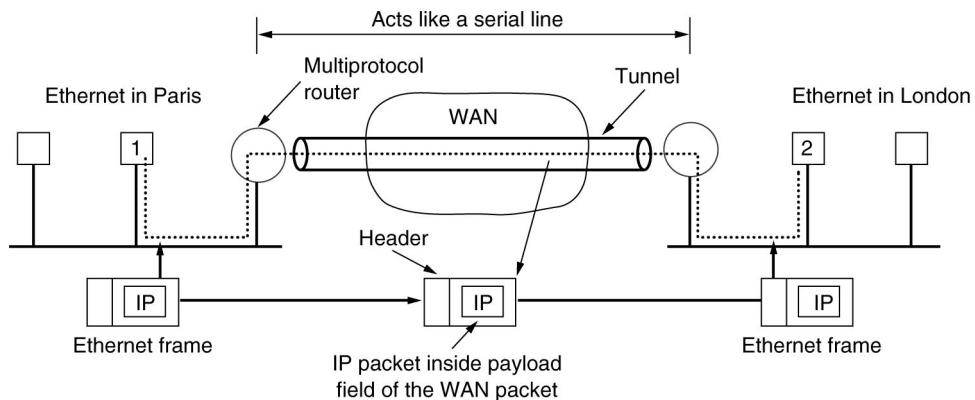
Las propiedades de este modelo, son las mismas que posee una sola SUBRED BASADA EN DATAGRAMAS:

- el único servicio que ofrece la **CAPA DE RED** a la **CAPA DE TRANSPORTE** es la **CAPACIDAD de inyectar DATAGRAMAS** en la **SUBRED** y **ESPERAR QUE TODO FUNCIONE BIEN**.
- no se requiere que todos los **PAQUETES** que pertenecen a **una conexión** atravesen **la misma secuencia** de **PUERTAS DE ENLACE**, pues para cada **PAQUETE** se toma una decisión de **ENRUTAMIENTO independiente** considerando posiblemente el **TRÁFICO ACTUAL**
- se pueden usar **múltiples rutas** para los **PAQUETES** pertenecientes a **una misma conexión**, logrando un **mejor aprovechamiento del ANCHO DE BANDA**
- **no hay garantías** de que los **PAQUETES** llegará**n EN ORDEN** al **DESTINO**, si es que llegan
- **no es posible que un PAQUETE de una red, transite por otra subred diferente**
- un **mayor potencial de CONGESTIÓN**, pero también un **mayor potencial para adaptarse a él**
- **ROBUSTEZ ANTE FALLAS** en los **ENRUTADORES**
- **necesidad de ENCABEZADOS más GRANDES**

5.5.5 Entunelamiento

El caso general de lograr la interacción de dos redes totalmente diferentes es en extremo difícil.

Para el **caso específico** en el que **dos HOSTS** que están **en redes de la misma clase** necesitan **conectarse**, pero **hay una (o varias) red diferente en el medio**, existe una **técnica** llamada **ENTUNELAMIENTO**.



El **HOST 1** quiere **enviar un PAQUETE IP al HOST 2**, entonces:

- construye el **PAQUETE IP** que **contiene la DIRECCIÓN IP del HOST 2**
- inserta el **PAQUETE** en el **CAMPO DE DATOS** de una **TRAMA ETHERNET**
- envía la **TRAMA ETHERNET** al **ENRUTADOR MULTIPROTOCOLO** (de Paris)

El **ENRUTADOR MULTIPROTOCOLO** (de Paris):

- **recibe la TRAMA ETHERNET y la DEPOSITA en la LÍNEA ETHERNET** (que se dirige hacia Londres)

El **ENRUTADOR MULTIPROTOCOLO** (de Londres):

- **recibe de la LÍNEA ETHERNET la TRAMA ETHERNET** y de su **CAMPO DE DATOS** extrae el **PAQUETE IP**
- inserta el **PAQUETE IP** en el **CAMPO DE DATOS** de una **NUEVA TRAMA ETHERNET**
- envía la **NUEVA TRAMA ETHERNET** hacia el **HOST 2**

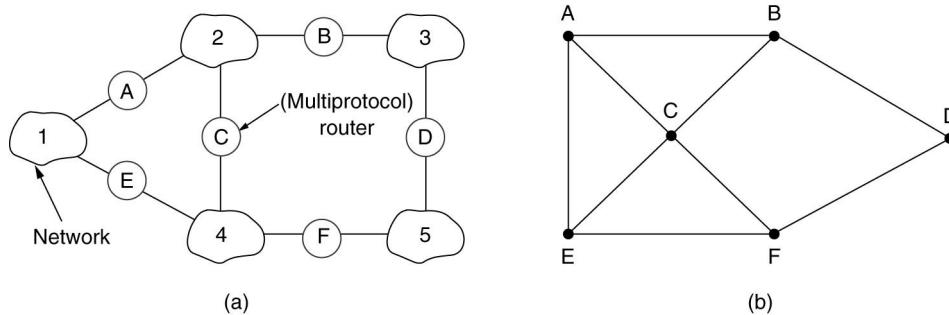
La **WAN** se **asemeja a un gran TÚNEL** que se extiende de un **ENRUTADOR MULTIPROTOCOLO** al otro.

El **PAQUETE IP** simplemente viaja desde un extremo del **TÚNEL** hacia el otro, no se preocupa por el **ENLACE WAN**.

Sólo el **ENRUTADOR MULTIPROTOCOLO** debe entender los **PAQUETES IP** y **WAN**. De hecho la **LÍNEA actúa como una LÍNEA SERIAL**

5.5.6 Enrutamiento entre redes

Es similar al **ENRUTAMIENTO** en una sola **SUBRED** pero con **algunas complicaciones adicionales**



La **INTERRED**, conformada por **REDES** interconectadas mediante **ENRUTADORES MULTIPROTOCOLO**, se representa mediante un **GRAFO**.

Pueden aplicarse los **ALGORITMOS DE ENRUTAMIENTO** conocidos, como **VECTOR DE DISTANCIA**, y el de **ESTADO DE ENLACE**, al **CONJUNTO DE ENRUTADORES MULTIPROTOCOLO**. Esto resulta en un **ALGORITMO DE ENRUTAMIENTO de DOS NIVELES**:

- **En la CAPA DE RED:**
Se utiliza un **PROTOCOLO DE PUERTA DE ENLACE INTERIOR (IGP)**
- Pero **entre ellas** (CAPA DE TRANSPORTE):
Se usa un **PROTOCOLO DE PUERTA DE ENLACE EXTERIOR (EGP)**

Puesto que **CADA RED** de una **INTERRED** es **independiente de las demás** con frecuencia se le llama **SISTEMA AUTÓNOMO (AS)**.

Estos **ALGORITMOS** de **ENRUTAMIENTO ENTRE LAS REDES** con frecuencia requieren que se crucen **FRONTERAS INTERNACIONALES**, por lo que entran en acción también las leyes de confidencialidad sobre la exportación de datos personales.

5.5.7 Fragmentación

Cada red impone un **tamaño máximo a sus PAQUETES**. Estos límites tienen varias razones, entre ellas:

- 1) El hardware (ejemplo: tamaño de una TRAMA ETHERNET)
- 2) El sistema operativo (ejemplos: todos los búferes son de 512 bytes)
- 3) Los protocolos (ejemplos: la cantidad de bits en el campo de longitud de paquete)
- 4) El cumplimiento de algún estándar (inter)nacional
- 5) El deseo de reducir hasta cierto nivel las retransmisiones inducidas por errores
- 6) El deseo de evitar que un paquete ocupe el canal demasiado tiempo

Estos y otros factores condicionan a los diseñadores, obteniéndose por ejemplo unas **cargas útiles máximas** como las siguientes:

- **48 bytes** en **CELDAS ATM**
- **65515 bytes** en **PAQUETES IP**

El problema surge cuando un **PAQUETE GRANDE** necesita atravesar una **RED DIFERENTE** que posee un **tamaño de PAQUETE demasiado pequeño**. En este caso la **solución es permitir** que las **PUERTAS DE ENLACE DIVIDAN los PAQUETES EN FRAGMENTOS**, enviando **cada FRAGMENTO como PAQUETE INTERRED INDIVIDUAL**.

Hay dos estrategias de fragmentación:

- **FRAGMENTACIÓN TRANSPARENTE:**

Es causada por una red de “**PAQUETES PEQUEÑOS**”. Los **PAQUETES** que llegan a dicha red y se dividen en **FRAGMENTOS**. Los **FRAGMENTOS** atraviesan la red y finalmente llegan todos ellos a la misma **PUERTA DE ENLACE** para ser **RECOMBINADOS**, obteniéndose nuevamente el **PAQUETE ORIGINAL**.

Parece simple pero **tiene problemas**, entre ellos:

- La **PUERTA DE ENLACE DE SALIDA** debe saber cuándo ha recibido todas las piezas, por lo que debe incluirse un **CAMPO de CONTEO o FIN DE PAQUETE, EN CADA PAQUETE**
- todas las piezas de un **PAQUETE** debe salir por la misma **PUERTA DE ENLACE DE SALIDA**, causando sobrecarga requerida para reensamblar los fragmentos (y volver a fragmentar)

- **FRAGMENTACIÓN NO TRANSPARENTE:**

Se trata de **abstener de recombinar** los **FRAGMENTOS** en las **PUERTAS DE ENLACE INTERMEDIAS**. Una vez fragmentado cada **PAQUETE**, sus fragmentos se tratan, y transitan atravesando las **PUERTAS DE ENLACES**, como si fuera el **PAQUETE ORIGINAL**. La **RECOMBINACIÓN** ocurre sólo en el **HOST de DESTINO. IP FUNCIONA DE ESTA FORMA**.

Los **problemas que tiene** son:

- la necesidad de que todos los **HOST** sean **CAPACES DE HACER EL REENSAMBLE**
- al fragmentarse un **PAQUETE GRANDE**, se añade **CARGA ADICIONAL** a la **RED**, pues cada fragmento posee su propio **ENCABEZADO**, sin embargo la posibilidad de usar varias **PUERTAS DE ENLACE ALTERNATIVAS** permite mejorar el desempeño
- Al fragmentar un **PAQUETE**, cada **FRAGMETO** debe **NUMERARSE** de tal manera que el **FLUJO DE DATOS ORIGINAL** pueda reconstruirse (ejemplo: usando un árbol). Las retransmisiones son perjudiciales para el sistema de numeración.

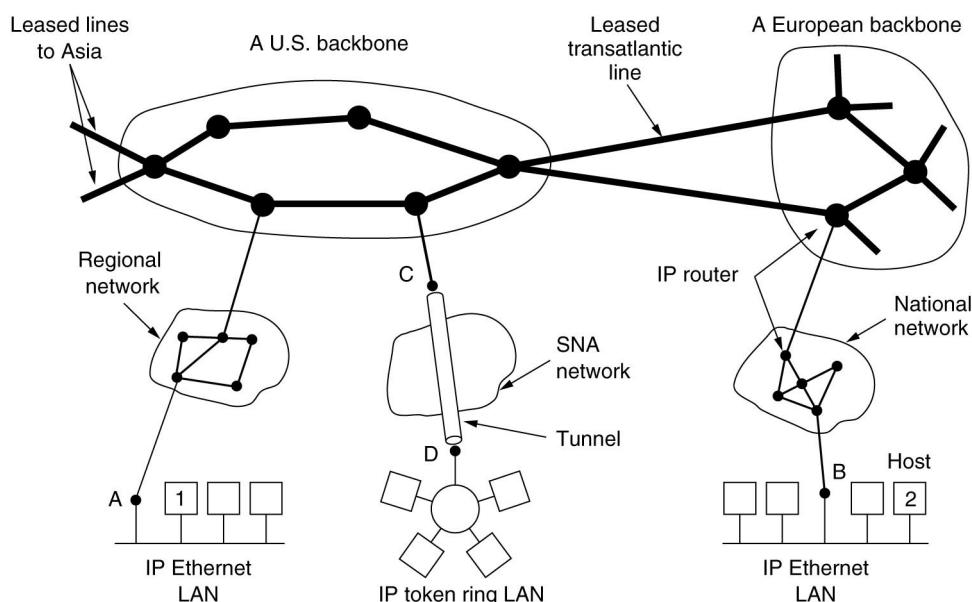
5.6 LA CAPA DE RED DE INTERNET

Los principios que guiaron el diseño de INTERNET, e hicieron posible su éxito se enumeran y analizan en el RFC 1958 (Clark, 1988 y Saltzer y cols., 1984). Entre ellos los 10 mejores principios son los siguientes:

- 1) ASEGÚRESE DE QUE FUNCIONA**
- 2) MANTENGA LA SIMPLICIDAD**
- 3) ELIJA OPCIONES CLARAS**
- 4) EXPLOTE LA MODULARIDAD**
- 5) PREVEA LA HETEROGENEIDAD**
- 6) EVITE LAS OPCIONES Y PARÁMETROS ESTÁTICOS**
- 7) BUSQUE UN BUEN DISEÑO; NO ES NECESARIO QUE SEA PERFECTO**
- 8) SEA ESTRICTO CUANDO ENVÍE Y TOLERANTE CUANDO RECIBA**
- 9) PIENSE EN LA CAPACIDAD DE CRECIMIENTO**
- 10) CONSIDERE EL DESEMPEÑO Y EL COSTO**

La **CAPA DE RED DE INTERNET** se puede ver como un **CONJUNTO DE SUBREDES o SISTEMAS AUTÓNOMOS INTERCONECTADOS**. Existen **varias REDES DE DORSALES PRINCIPALES** construidas mediante **LÍNEAS CON ALTO ANCHO DE BANDA** y **ENRUTADORES RÁPIDOS**, que están **conectadas a varias REDES REGIONALES** de nivel medio, que a su vez están **conectadas a varias REDES LANs** de muchas **UNIVERSIDADES, COMPAÑÍAS y PROVEEDORES DE SERVICIOS DE INTERNET (ISP)** formando un **DISEÑO CUASIJERÁRQUICO**.

Lo que mantiene unida a **INTERNET** es el **PROTOCOLO IP: PROTOCOLO DE INTERNET**, diseñado desde un principio para la **INTERCONEXIÓN DE REDES DIFERENTES**. Proporciona un **MÉTODO DE MEJOR EFUERZO SIN GARANTÍAS** para el **TRANSPORTE de DATAGRAMAS** desde el **ORIGEN** hasta el **DESTINO**.



La **COMUNICACIÓN EN INTERNET** funciona como sigue:

- **CAPA DE TRANSPORTE** toma **FLUJOS de DATOS** y **los divide** en **DATAGRAMAS** de 64 Kbytes (1500 Kbytes en la práctica por lo que se ajustan a TRAMAS ETHERNET, de la CAPA DE ENLACE)
- **Cada DATAGRAMA** se **transmite a través de INTERNET** posiblemente **FRAGMENTÁNDOSE** en **unidades más pequeñas**, durante el camino a la

máquina DESTINO

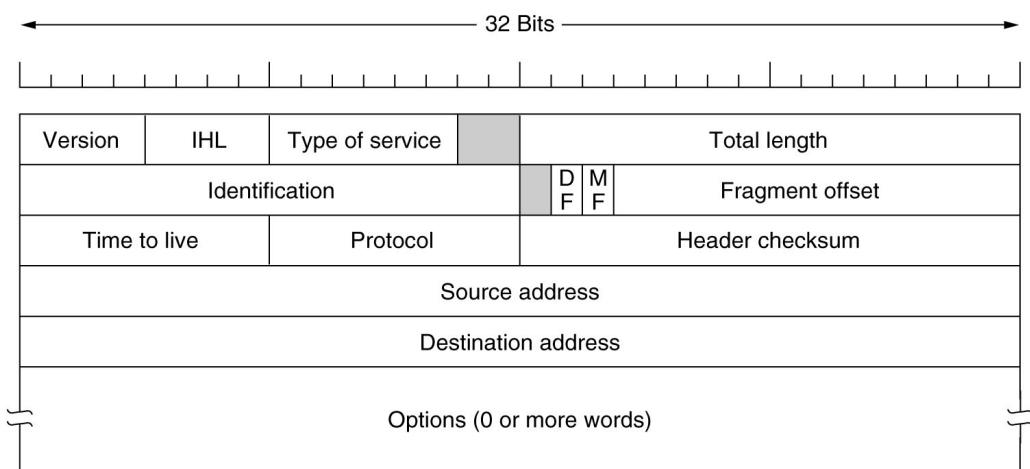
- Finalmente **llegan todos los FRAGMENTOS al HOST DESTINO, donde:**
 - LA CAPA DE RED REENSAMBLA** todos los **FRAGMENTOS** OBTENIENDO **EL DATAGRAMA ORIGINAL**.
 - El DATAGRAMA se entrega a la CAPA DE TRANSPORTE** obteniendo el **FLUJO DE ENTRADA PARA EL PROCESO RECEPTOR**.

5.6.1 El protocolo IP

Un DATAGRAMA de IP consta de una:

- PARTE DE ENCABEZADO**, de longitud variable, que posee:
 - PARTE FIJA** de 20 bytes
 - PARTE OPCIONAL** de longitud variable
- PARTE DE TEXTO**

El ENCABEZADO de un DATAGRAMA IP tiene el siguiente FORMATO:



El **ENCABEZADO** de un **DATAGRAMA IP**:

- se transmite en **ORDEN BIG ENDIAN**: de izquierda a derecha, comenzando con el **bit de orden mayor del CAMPO VERSIÓN** (**SPARC** usa **BIG ENDIAN**, **INTEL** usa **LITTLE ENDIAN** por lo que debe hacerse una conversión por software)
- consta de los siguientes CAMPOS**
 - VERSIÓN**
 - IHL**
 - TIPO DE SERVICIO**
 - LONGITUD TOTAL**
 - IDENTIFICACIÓN**
 - DF**
 - MF**
 - DEPLAZAMIENTO DEL FRAGMENTO**
 - TIEMPO DE VIDA**
 - PROTOCOLO**
 - SUMA DE VERIFICACIÓN DEL ENCABEZADO**
 - DIRECCIÓN DE ORIGEN**
 - DIRECCIÓN DE DESTINO**
 - OPCIONES**

La función que cumple cada uno de estos campos se detalla a continuación:

CAMPOS DEL ENCABEZADO DE UN DATAGRAMA IP	FUNCIÓN
VERSIÓN	Versión del PROTOCOLO IP
IHL	Indica la longitud del encabezado
TIPO DE SERVICIO	Distinguir entre diferentes clases de servicios Los enrutadores actuales lo ignoran
LONGITUD TOTAL	Tamaño de todo el datagrama: encabezado + datos
IDENTIFICACIÓN	Identifica a qué DATAGRAMA pertenece uno de sus fragmentos Todos los fragmentos de un datagrama tienen el mismo valor de identificación
DF	Don't fragment: No fragmentar Ejemplo: al arrancar un host su ROM (BIOS) podría pedir el envío de una imagen binaria de memoria como un solo datagrama
MF	More fragments: Más fragmentos Todos los fragmentos excepto el último tienen establecido este bit
DESPLAZAMIENTO DEL FRAGMENTO	Indica en qué parte del datagrama actual va el fragmento. La unidad de fragmento fundamental mide 8 bytes.
TIEMPO DE VIDA	Contador que limita la vida de un paquete decrementándose en cada salto (cada vez que atraviesa un enrutador o host)
PROTOCOLO	Indica el protocolo de las capas superiores al que debe entregarse el paquete. TCP es una posibilidad, pero también está UDP y algunos más.
SUMA DE VERIFICACIÓN DEL ENCABEZADO	Verifica al encabezado Útil para la detección de errores
DIRECCIÓN DE ORIGEN	La dirección IP, indica: número de red + número de host
DIRECCIÓN DE DESTINO	
OPCIONES	Proporciona un recurso para incluir más información necesaria para futuras versiones del protocolo IP Originalmente se definieron cinco opciones: <ul style="list-style-type: none"> - Seguridad: Especifica que tan secreto es el datagrama - Enrutamiento estricto desde origen: indica la ruta completa a seguir - Enrutamiento libre desde el origen: da una lista de enrutadores que no deben evitarse - Registrar ruta: Hace que cada enrutador agregue su dirección IP - Marca de tiempo: Hace que cada enrutador agregue su dirección IP y su marca de tiempo

5.6.2 Direcciones IP

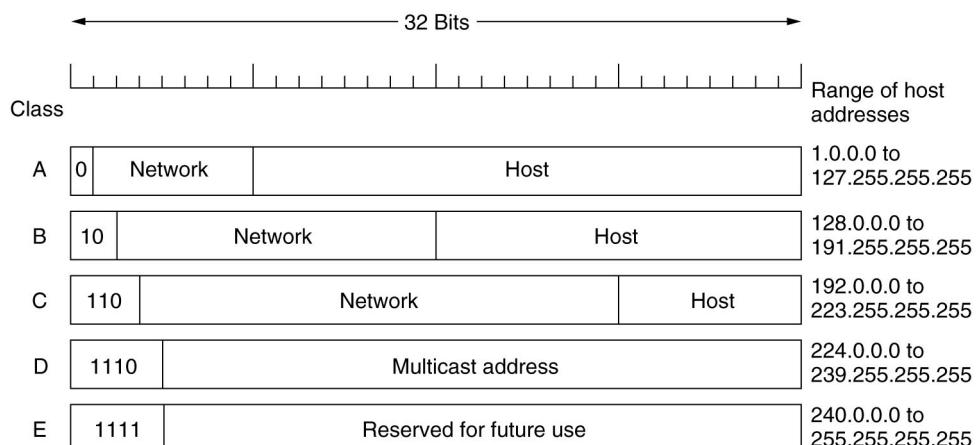
Cada **HOST** y cada **ENRUTADOR** de **INTERNET** tienen una **DIRECCIÓN IP**, que **codifica SU NÚMERO DE RED y SU NÚMERO DE HOST**.

La **combinación es única**: no hay dos máquinas que tengan la misma dirección IP.

Todas las direcciones IP son de 32 bits de longitud(en Ipv4).

En realidad una dirección IP no se refiere a un host, sino mas bien a una **INTERFAZ DE RED** que se encuentra instalada en el **HOST**, por lo tanto si el **HOST** se encuentra en dos redes, debe tener dos direcciones IP

Por décadas las **DIRECCIONES IP se dividieron en CINCO CLASES** conforme al **DIRECCIONAMIENTO CON CLASE (CLASSFULL ADDRESSING)**, que **actualmente no se usa más:**



Los **números de redes son manejados por** una corporación no lucrativa llamada **ICANN (CORPORACIÓN DE INTERNET PARA LA ASIGNACIÓN DE NOMBRES Y NÚMEROS)** para evitar conflictos.

Las direcciones de red **generalmente se escriben en NOTACIÓN DECIMAL CON PUNTOS**, cada uno de los 4 bytes se escriben en decimal, de 0 a 255.

La dirección IP menor es 0.0.0.0

La dirección IP mayor es 255.255.255.255

0 0	This host
0 0 ... 0 0	A host on this network
1 1	Broadcast on the local network
Network 1 1 1 1 ... 1 1 1 1	Broadcast on a distant network
127 (Anything)	Loopback

Los valores:

- **0.0.0.0** significa **ESTA RED o ESTE HOST.**
- **255.255.255.255** significa la **DIRECCIÓN DE DIFUSIÓN (BROADCAST) EN ESTA RED LOCAL**
-
- **[RED=0...0, HOST=x...x]** denota **un HOST EN ESTA RED LOCAL**
- **[RED=x...x, HOST=1...1]** denota **LA DIRECCIÓN DE DIFUSIÓN (BROADCAST) en una RED REMOTA DISTANTE**
- **127.xxx.xxx.xxx** denota **direcciones locales de prueba (loopbacks)**

Todos los HOSTS de una RED deben tener el MISMO NÚMERO DE RED.

Subredes

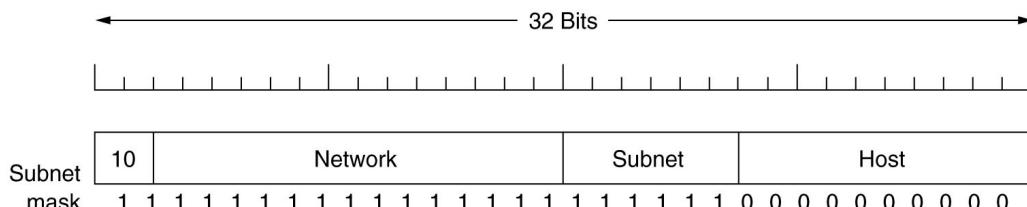
Esta forma de DIRECCIONAMIENTO IP puede causar problemas conforme las REDES crecen.

Las DIRECCIONES IP son cada vez más escasas, por lo que NO ES FÁCIL conseguir DIRECCIONES IP ADICIONALES.

Por lo tanto, las organizaciones optaron por **DIVIDIR SUS REDES EN PORCIONES LLAMADAS SUB-REDES DE USO INTERNO**, que **en conjunto actúan como una sola RED LOCAL.**

Para implementar las **SUB-REDES** el **ENRUTADOR PRINCIPAL**, que está **conectado al ISP**, necesita **UNA MÁSCARA DE SUB-RED** que **indica la división entre:**

NÚMERO DE RED + NÚMERO DE SUB-RED + NÚMERO DE HOST



Una RED CLASE B dividida en 64 SUB-REDES

Fuera de la RED, las SUB-REDES no son visibles, por lo que la asignación de una SUB-RED no requiere comunicación con el ICANN ni modificar bases de datos externas.

Cuando llega un **PAQUETE IP** a un **ENRUTADOR**:

- busca su **DIRECCIÓN DESTINO** en la **TABLA DE ENRUTAMIENTO**
- Si el **PAQUETE** es:
 - para un **HOST** en una **RED DISTANTE**:
Lo reenvía al siguiente **ENRUTADOR** de la **INTERFAZ** dada en la **TABLA**
 - para un **HOST** de la **RED LOCAL**:
Lo envía directamente al **HOST DESTINO**
 - **inexistente en la TABLA DE ENRUTAMIENTO**:
Lo reenvía al **ENRUTADOR PREDETERMINADO** que **POSEE TABLAS MÁS EXTENSAS**

Las **ENTRADAS EN LA TABLA DE ENRUTAMIENTO** tienen la forma

(tal RED, tal SUB-RED, tal HOST)

y todo lo que necesita hacer **EL ENRUTADOR** es una **OPERACIÓN** de **AND BOOLEANO** entre:

DIRECCIÓN IP DESTINO

y

MÁSCARA DE SUB-RED,

para deshacerse del **NÚMERO DE HOST**, y **BUSCAR** la **DIRECCIÓN RESULTANTE** en **SUS TABLAS**.

CIDR – Enrutamiento interdominios sin clases

El direccionamiento IP por clases desperdicia millones de direcciones IP.

La solución que se implementó es el **CIDR (Enrutamiento interdominios sin clases)** que se describe en el **RFC 1519**.

Su concepto base es asignar las direcciones IP restantes en bloques de tamaño variable. Este esquema sin clases hace más complicado el reenvío.

Posee una **NOTACIÓN** similar a la **NOTACIÓN DECIMAL CON PUNTOS**, pero adiciona una “barra diagonal” (/), seguida de un número que indica la cantidad de bits en uno que posee la máscara de SUB-RED contando de izquierda a derecha.

En cada **ENRUTADOR** las **ENTRADAS EN SUS TABLAS DE ENRUTAMIENTO** se **AMPLÍAN PARA CONTENER** la **MÁSCARA DE SUB-RED (de 32 bits)**.

De esta manera una **ENTRADA** típica en la **TABLA DE ENRUTAMIENTO** consta de tres factores:
(DIRECCIÓN IP , MÁSCARA DE SUBRED , LÍNEA SALIENTE)

Cuando **Ilega** un **PAQUETE IP** al **ENRUTADOR**:

- Se **extrae** su **DIRECCIÓN IP DE DESTINO**
- Analiza la **TABLA DE ENRUTAMIENTO**, **ENTRADA** por **ENTRADA**,
 - **haciendo la operación lógica AND Booleano DIRECCIÓN IP DE DESTINO & MÁSCARA DE SUB-RED**
 - **comparando la DIRECCIÓN RESULTANTE** con **LA DIRECCIÓN IP DE ESA ENTRADA EN LA TABLA DE ENRUTAMIENTO**
Si hay múltiples coincidencias, se elige aquella en la que interviene la **MÁSCARA DE SUB-RED CON MÁS BITS ESTABLECIDOS EN UNO** (ejemplo entre /20 y /24 elige /24)

Se han diseñado muchos algoritmos complejos para acelerar el proceso de coincidencia de DIRECCIÓN, y actualmente existen **ENRUTADORES COMERCIALES VLSI** programados con estos algoritmos.

NAT – Traducción de dirección de red

El **problema de quedarse sin direcciones IP** no es un problema teórico, **está sucediendo aquí y ahora mismo**.

La **solución a largo plazo** es **migrar, todo INTERNET, al PROTOCOLO IPv6**, que tiene **direcciones de 128 bits**.

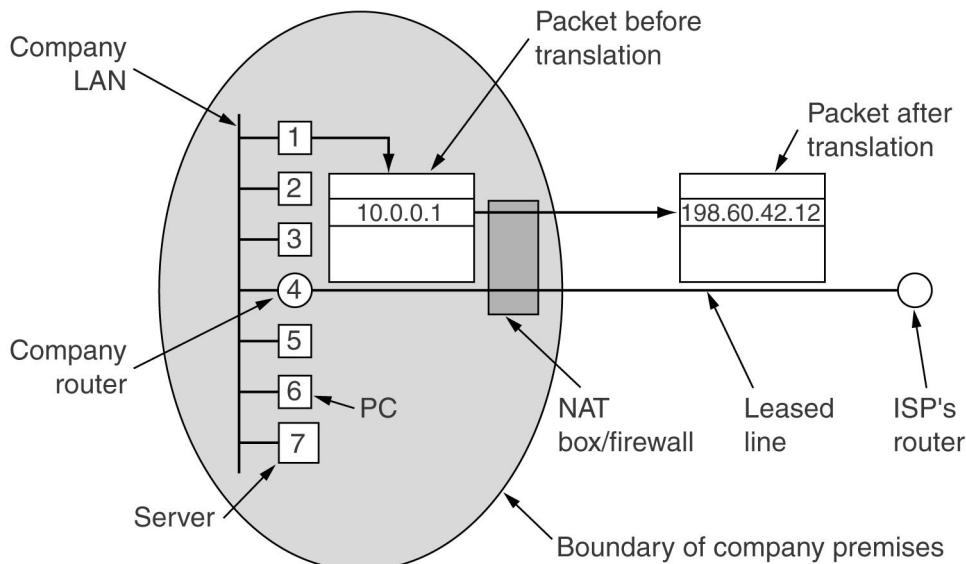
Pero **una solución (no muy elegante)** que se implementó **a corto plazo** surgió en la forma de **TRADUCCIÓN DE DIRECCIÓN DE RED (NAT) y se describe en el RFC 3022**.

La idea es **ASIGNAR UNA SOLA DIRECCIÓN IP (o un grupo pequeño) A CADA ORGANIZACIÓN**.

DENTRO DE LA ORGANIZACIÓN, cada computadora tiene una **DIRECCIÓN IP ÚNICA** que se usa para **ENRUTAR** el **TRÁFICO INTERNO** y puede poseer un valor dentro de los siguientes **RANGOS DE DIRECCIONES DE RED PRIVADAS**:

10.0.0.0/8	- 10.255.255.255/8	(16.777.216 HOSTS)
172.16.0.0/12	- 172.31.255.255/12	(1.048.576 HOSTS)
192.168.0.0/16	- 192.168.255.255/16	(65.536 HOSTS)

Además la **ORGANIZACIÓN** debe disponer de una **CAJA NAT** que convierte la **DIRECCIÓN IP INTERNA DEL HOST ORIGEN** a la **VERDADERA DIRECCIÓN IP DE LA ORGANIZACIÓN EN INTERNET**, como se muestra:



A menudo, la **CAJA NAT** se combina en un solo dispositivo con un **FIREWALL (SERVIDOR DE SEGURIDAD)** que controla cuidadosamente todo lo que entra y lo que sale de la **ORGANIZACIÓN**.

El problema de **NAT** es que **utiliza información de la CAPA SUPERIOR, LA CAPA DE TRANSPORTE**, para poder identificar a qué host de la **RED LOCAL INTERNA** se debe enrutar un **mensaje de RESPUESTA QUE PROVIENE DESDE INTERNET**, en base a los **NÚMEROS DE PUERTO DE ORIGEN y DE DESTINO** que se encuentran en los **CAMPOS DE CARGA ÚTILES** de TCP o de UDP.

Esto rompe totalmente con el modelo de CAPAS INDEPENDIENTES haciendo **VULNERABLE** a **NAT** ante cambios en los **PROTOCOLOS TCP o UDP**.

Los puertos **0 a 1023** están **reservados para los servicios bien conocidos** (ejemplo: **80 es el usado por los servidores Web**).

Cuando un **PAQUETE IP** llega a la **CAPA NAT** desde el **ISP**:

- Extrae el **PUERTO DE ORIGEN** del **ENCABEZADO TCP (o UDP)**
- Lo **búsqueda en la TABLA DE TRADUCCIÓN NAT**, para encontrar la **DIRECCIÓN IP INTERNA**
- Desde la **ENTRADA LOCALIZADA** se **extraen e insertan en un PAQUETE IP** la **DIRECCIÓN IP INTERNA** y el **PUERTO TCP (o UDP) DE ORIGEN**. Las **SUMAS DE VERIFICACIÓN IP y TCP(o UDP)** se **recalculan e insertan en el PAQUETE**
- Envía el **PAQUETE** al **ENRUTADOR de la ORGANIZACIÓN** para su entrega normal

Las **objeciones a NAT** son las siguientes (mas detalles en el **RFC 2993**):

- viola el **MODELO ARQUITECTÓNICO DEL PROTOCOLO IP** que establece que cada dirección IP identifica a un host único globalmente
- causa que **INTERNET cambie de una RED SIN CONEXIÓN a una RED ORIENTADA A LA CONEXIÓN**, con la **vulnerabilidad que eso implica si la CAJA NAT sufre fallas**
- **viola la regla más fundamental de los PROTOCOLOS DE CAPAS:** la **CAPA K no puede hacer suposiciones** de lo que **suceda en la CAPA K+1**

5.6.3 Protocolos de Control en Internet

El **PROTOCOLO IP** se usa para transferir los datos.

INTERNET tiene también **PROTOCOLOS DE CONTROL**, que se usan en la **CAPA DE RED** como: **ICMP, ARP, RARP, BOOTP, DHCP**

Protocolo de Mensajes de Control de Internet – ICMP

Los **ENRUTADORES** supervisan el **funcionamiento de INTERNET**. Cuando **ocurre algo inesperado ICMP informa de un evento**. También **se utiliza para diagnosticar fallas y probar el funcionamiento de INTERNET**.

Los mensajes se **encapsulan** en un **PAQUETE IP**.

TIPO DE MENSAJE	DESCRIPCIÓN
Destination unreachable	El paquete no se puede entregar
Time exceeded	Campo de tiempo de vida = 0
Parameter problem	Campo de encabezado inválido
Source quench	Paquete regulador
Redirect	Se usa cuando en enrutador se percata de que un paquete está mal enrutado
Echo	Pregunta a una máquina si está viva
Echo reply	Respuesta de host en actividad
Timestamp request	Solicitud de eco con marca de tiempo
Timestamp reply	Respuesta a solicitud de eco con marca de tiempo

Principales tipos de mensajes ICMP

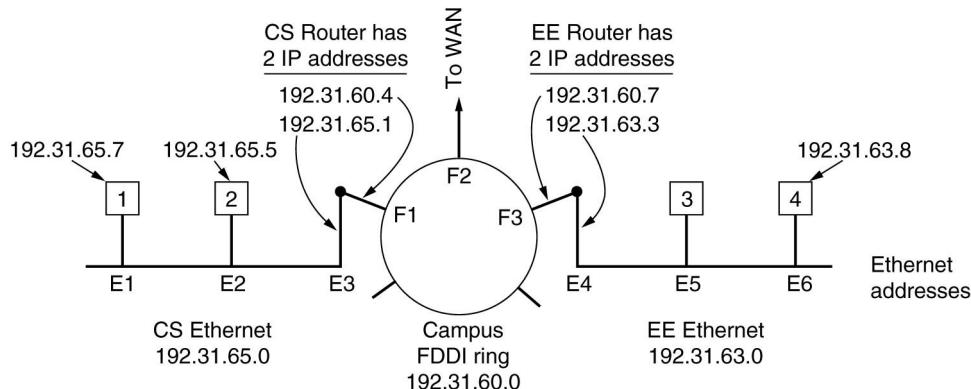
La lista completa en <http://www.iana.org/assignments/icmp-parameters>

ARP – Protocolo de Resolución de Direcciones

Aunque en **INTERNET** cada **HOST o ENRUTADOR** tiene **una (o más) DIRECCIONES IP**, éstas **no pueden usarse para enviar paquetes** porque el **HARDWARE de CAPA DE ENLACE DE DATOS NO ENTIENDE las DIRECCIONES DE INTERNET**, sólo entiende direcciones de la

CAPA DE ENLACE DE DATOS, como las DIRECCIONES MAC (de 48 bits) de las REDES ETHERNET.

Debe haber un modo de asociar y convertir DIRECCIONES IP en DIRECCIONES DE LA CAPA DE ENLACE (MAC de ETHERNET).



Un usuario del **HOST 1** quiere enviar un **PAQUETE** hacia el **HOST 2**. El **EMISOR** sabe el **nombre del receptor: mary@eagle.cs.uni.edu**

Hay que encontrar la **DIRECCIÓN IP del host llamado eagle.cs.uni.edu**

Esta consulta la realiza el **SISTEMA DE NOMBRES DE DOMINIO (DNS)** que devuelve la DIRECCIÓN IP correspondiente al nombre de host.

- El **HOST 1** emite un **PAQUETE DE DIFUSIÓN** hacia la **RED ETHERNET** preguntando:

¿Quién posee la dirección IP 192.31.65.5?

Esta pregunta se lleva a cabo usando **ARP**

- La **DIFUSIÓN** llegará a cada **HOST de la RED** y cada uno **verificará su DIRECCIÓN IP**
- El **HOST 2 responde** con su **DIRECCIÓN ETHERNET: E2**
- El **HOST 1 aprende** que la **DIRECCIÓN IP: 192.31.65.5** están en el **HOST 2 con DIRECCIÓN ETHERNET: E2**

Usar ARP es mucho más ventajoso que usar archivos de configuración manuales que deben ser mantenidos por personal adecuado.

Una optimización de ARP es hacer que almacene temporalmente la información en una caché, para evitar una segunda difusión.

Otra mejora es hacer que **cada HOST difunda su CORRESPONDENCIA cuando arranca**, en este caso no debería haber respuesta, pues si la hubiere, significa que se ha asignado una misma **DIRECCIÓN IP** a dos **HOSTS DISTINTOS**, por lo tanto la última en inicializarse deberá informar al administrador de la red y no arrancar la interfaz de red.

RARP, BOOTP y DHCP

El problema a resolver es saber cuál es la **DIRECCIÓN IP** de tal **DIRECCIÓN DE CAPA DE ENLACE DE DATOS** (**MAC en REDES ETHERNET**).

La **primera solución** inventada fue el **PROTOCOLO DE RESOLUCIÓN DE DIRECCIÓN DE RETORNO (RARP)**, definido en el **RFC 903**. Permite que un **HOST** recientemente inicializado transmita su **DIRECCIÓN ETHERNET** en un mensaje como:

Mi DIRECCIÓN ETHERNET de 48 bits es 14:04:05:18:01:25

¿Alguien sabe cuál es mi DIRECCIÓN IP?

y devuelve la **DIRECCIÓN IP CORRESPONDIENTE**

La **desventaja de ARP** es que **para enviar los mensajes** usa **DIRECCIONES DE DIFUSIÓN**, éstas **no las envían los ENRUTADORES** por lo que **se requiere un SERVIDOR RARP en cada RED LOCAL**

Para resolver esto, se diseño un **PROTOCOLO DE ARRANQUE ALTERNATIVO** llamado **BOOTP**, que usa **PAQUETES UDP**, los **cuáles** pueden **enviarse a través de los ENRUTADORES**, y que **proporciona información adicional a HOSTS SIN DISCOS**, como la **DIRECCIÓN IP DEL SERVIDOR DE ARCHIVOS** que contiene la **IMAGEN DE MEMORIA**, la **DIRECCIÓN IP DEL ENRUTADOR PREDeterminado** y la **MÁSCARA DE SUB-RED**. Se describe en los **RFCs 951, 1048 y 1084**.

Problema serio de BOOTP, requiere configuración manual y supervisión humana.

Por este motivo, **BOOTP se extendió** y se le dio un nombre nuevo: **DHCP (PROTOCOLO DE CONFIGURACIÓN DE HOST DINÁMICO)** y **permite una asignación de DIRECCIONES IP MANUAL y AUTOMÁTICA**. Se describe en los **RFCs 2131 y 2132**. También se basa en la **existencia de un SERVIDOR ESPECIAL**, aunque en este caso **NO SE NECESITA QUE ESTÉ en la misma LAN que el HOST solicitante** (las conexiones a un ISP mediante líneas ADSL usan esta característica).

Un **problema** que se presenta es **por cuánto tiempo debe asignarse una DIRECCIÓN IP**. Si un **HOST** deja la **RED** y no devuelve su **DIRECCIÓN IP** al **SERVIDOR DHCP**, ésta se pierde. Para impedir esto la **asignación de la DIRECCIÓN IP posee un período fijo de validez**. Luego, antes de que expire ese tiempo, el **HOST** debe renovar su **DIRECCIÓN IP** con el **SERVIDOR DHCP**.

5.6.4 OSPF – Protocolos de enrutamiento de puerta de enlace interior

En **ARPANET** se usó un **PROTOCOLO DE VECTOR DE DISTANCIAS (RIP)**

La **IETF** empezó a trabajar en **un sucesor en 1979**, llamado **OSPF (ABRIR PRIMERO LA RUTA MÁS CORTA)**. En **1990 se convirtió en NORMA**. Se describe en el **RFC 2328**.

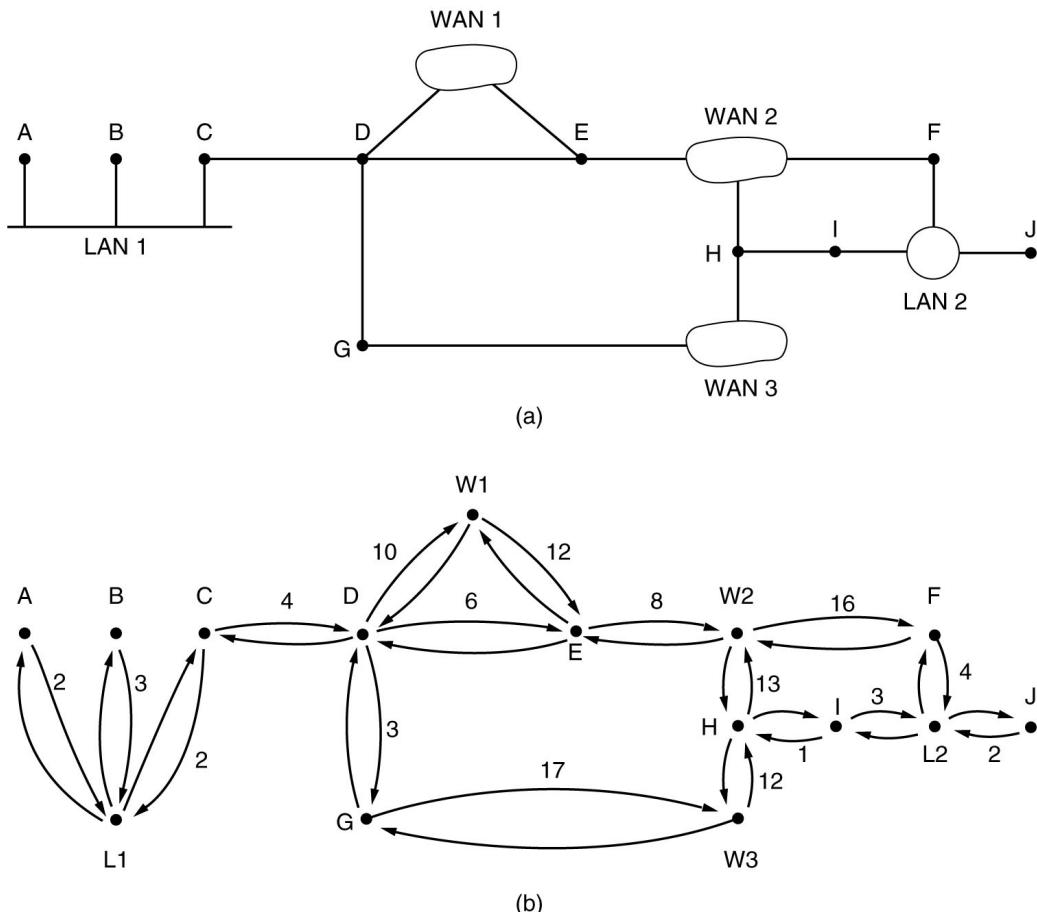
Los requisitos que tenía que cumplir:

- el algoritmo se tenía que publicar en la literatura abierta
- tenía que apoyar una gran variedad de métrica de distancia
- debía ser dinámico y adaptable rápidamente a los cambios de topología
- debía soportar el enrutamiento en base al tipo de servicio
- debía soportar el balanceo de carga, dividiéndola en múltiples rutas
- debía soportar los sistemas jeráquicos
- debía poseer una pizca de seguridad
- debía prever la situación de los enrutadores que se conectaban a INTERNET a través de un TÚNEL

OSPF soporta tres tipos de **CONEXIONES** y **REDES**:

- La líneas **PUNTO a PUNTO** entre **dos ENRUTADORES**
- **Redes multiacceso con DIFUSIÓN** (ejemplo: mayoría de las **LANs**)
Una red multiacceso tiene muchos enrutadores que se conectan y comunican directamente entre sí y con todos los demás.
- **Redes multiacceso sin DIFUSIÓN**
(ejemplo: la mayoría de las **WANs de paquetes conmutados**)

OSPF funciona **resumiendo** la **COLECCIÓN de REDES, ENRUTADORES y LÍNEAS en un GRAFO DIRIGIDO PONDERADO** (cada ARCO tiene un COSTO). Conexión serie entre dos ENRUTADORES se representa como UN PAR DE ARCOS entre dos NODOS, uno en cada dirección y cuyos pesos pueden ser distintos.



OSPF divide los **SISTEMAS AUTÓNOMOS** en **ÁREAS**.

Un **ÁREA** es una **RED DORSAL** o un **CONJUNTO DE REDES INMEDIATAS**.

Cada **SISTEMA AUTÓNOMO** tiene un **ÁREA DE RED DORSAL**. Todas las otras **ÁREAS** se conectan a la **RED DORSAL**, posiblemente por **TÚNELES**. Desde un **ÁREA** puede accederse mediante la **RED DORSAL** a cualquier otra **ÁREA** dentro del mismo **SISTEMA AUTÓNOMO**. Cada **ENRUTADOR** que se conecta a dos o más **ÁREAS** forma parte de la **RED DORSAL**.

Dentro de una **ÁREA**, cada **ENRUTADOR** posee la **misma base de datos de ESTADO DE ENLACE** y ejecuta el mismo **ALGORITMO de ENRUTAMIENTO** de la **RUTA MÁS CORTA HACIA CUALQUIER OTRO ENRUTADOR dentro de la misma ÁREA**.

Los **ENRUTADORES** de la **RED DORSAL** necesitan tantas **base de datos de ESTADO DE ENLACE** como **ÁREAS** a las que están conectados y requieren usar más de un **ALGORITMO de ENRUTAMIENTO por TURNOS**.

Durante la operación normal **pueden necesitarse tres tipos de rutas**:

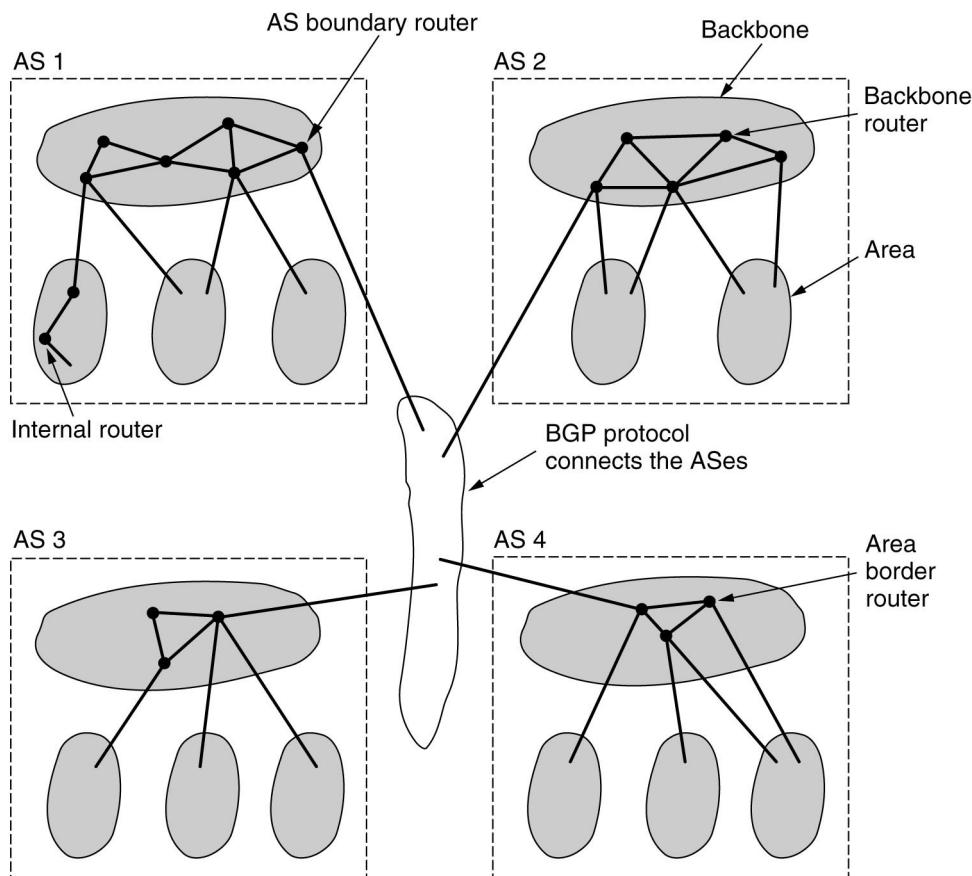
- **dentro del ÁREA**
- **entre ÁREAS**
- **entre SISTEMAS AUTÓNOMOS**

El **ENRUTAMIENTO ENTRE ÁREAS** siempre procede en tres pasos:

- 1. Va del ÁREA de origen a la RED DORSAL**
- 2. Va a través de la RED DORSAL hacia el ÁREA DESTINO**
- 3. Va al Área de DESTINO**

Esto implica una **configuración en estrella**, donde:

- el **CONCENTRADOR** es la **RED DORSAL**
- los **extremos terminales** son **cada una** de las **restantes ÁREAS** del **SISTEMA AUTÓNOMO**



OSPF distingue **cuatro clases de ENRUTADORES**:

1. **ENRUTADORES INTERNOS** que están **totalmente dentro de un ÁREA**
2. **ENRUTADORES DE LÍMITE DE ÁREA** que **conectan dos o más ÁREAS**
3. **ENRUTADORES DE LA RED DORSAL** que están **en la RED DORSAL**
4. **ENRUTADORES FRONTERIZOS DE SISTEMAS AUTÓNOMOS** que se **comunican con los ENRUTADORES DE OTROS SISTEMAS AUTÓNOMOS**

Éstas clases se pueden traslapar.

OSPF trabaja intercambiando **información entre ENRUTADORES ADYACENTES** que no es lo mismo que entre **ENRUTADORES VECINOS**. Para esto elige un **ENRUTADOR** como **ENRUTADOR DESIGNADO**. Se dice que es **ADYACENTE** a todos los demás **ENRUTADORES** en su **LAN** e intercambia información con ellos. Los **ENRUTADORES VECINOS** que **NO SON ADYACENTES NO INTERCAMBIAN INFORMACIÓN ENTRE SÍ**.

Los **MENSAJES ENTRE ENRUTADORES** se **DIFUNDEN** mediante **INUNDACIÓN**:

TIPO DE MENSAJE	DESCRIPCIÓN
Hello	Descubre quienes son los vecinos
Link state update	Proporciona los costos del emisor a sus vecinos
Link state ack	Confirma la recepción de la actualización del estado del enlace
Database description	Anuncia qué actualizaciones tiene el emisor
Link state request	Solicita información del socio

5.6.5 BGP – Protocolo de Puerta de Enlace de Frontera

Se describe en los **RFCs 1771 a 1774**.

Se usa en los **ENRUTADORES** que **comunican dos o más SISTEMAS AUTÓNOMOS**.

En general estos **ENRUTADORES** del **PROTOCOLO DE PUERTA DE ENLACE DE FRONTERA** tienen que **preocuparse por cuestiones políticas**.

Estas **REGLAS POLÍTICAS**, en **CADA ENRUTADOR BGP** se configuran **manualmente** (o usando algún tipo de escritura) y **NO SON PARTE DEL PROTOCOLO**.

Dos o más SISTEMAS AUTÓNOMOS se consideran **CONECTADOS** si hay una **LÍNEA** entre **LOS ENRUTADORES FRONTERIZOS** de cada uno

BGP clasifica las **REDES** en base al **TRÁFICO** en tres categorías:

- **REDES STUB:**
Tienen **sólo una conexión** en el GRAFO de BGP
No se pueden usar para transportar tráfico porque no hay nadie del otro lado.
- **REDES MULTICONECTADAS:**
Podrían usarse para transportar tráfico, salvo que se indique lo contrario.
- **REDES DE TRÁNSITO:**
Redes DORSALES que están **dispuestas a ocuparse de PAQUETES DE TERCEROS, posiblemente con algunas restricciones, y normalmente por pagos.**

5.6.6 Multidifusión de Internet

Algunas aplicaciones requieren que **un PROCESO** pueda **enviar SIMULTÁNEAMENTE a una GRAN CANTIDAD DE RECEPTORES** (ejemplo: actualizaciones de datos en una base de datos distribuida).

El **PROTOCOLO IP** apoya la **MULTIDIFUSIÓN**, usando **DIRECCIONES IP de CLASE D: [224.0.0.0 – 239.255.255.255]**

Se soportan **dos TIPOS DE DIRECCIONES DE GRUPO:**

- **PERMANENTES:**

Siempre están allí y no se tiene que preparar

Ejemplos:

224.0.0.1 TODOS LOS SISTEMAS EN UNA LAN

224.0.0.2 TODOS LOS ENRUTADORES EN UNA LAN

224.0.0.5 TODOS LOS ENRUTADORES DE OSPF EN UNA LAN

224.0.0.6 TODOS LOS ENRUTADORES DESIGNADOS DE OSPF EN UNA LAN

- **TEMPORALES:**

Primero deben ser creados. Un **PROCESO** puede pedir a su **HOST** que se una a un **GRUPO ESPECÍFICO** (también que lo abandone). Cuando el último **PROCESO** en un **HOST** deja un **GRUPO**, ese **GRUPO** ya no está presente en el **HOST**.

La **MULTIDIFUSIÓN** se implementa mediante **ENRUTADORES DE MULTIDIFUSIÓN ESPECIALES**, que pueden colocarse o no, con **ENRUTADORES NORMALES**.

Los **PAQUETES DE PREGUNTA** y **RESPUESTAS** utilizan un **PROTOCOLO** llamado **IGMP (PROTOCOLO DE ADMINISTRACIÓN DE GRUPO DE INTERNET)**. Es muy similar al **ICMP**. Tiene **sólo dos tipos de PAQUETES: PREGUNTA y RESPUESTA**, cada uno con **formato simple, fijo**, con **alguna información de control** en la primera palabra del **CAMPO DE CARGA ÚTIL** y una **DIRECCIÓN IP CLASE D** en la segunda palabra.

El **ENRUTAMIENTO DE MULTIDIFUSIÓN** se logra usando **ÁRBOLES DE DIFUSIÓN**.

IGMP hace gran uso de **TÚNELES** para **no molestar a los nodos que no pertenecen al ÁRBOL DE EXPANSIÓN**.

5.6.7 IP móvil

Muchos usuarios de **INTERNET** tienen **computadoras portátiles** y **desean permanecer conectados todo el tiempo** incluso mientras están de viaje.

El problema es el propio **ESQUEMA DE DIRECCIONAMIENTO**:

Número de RED + Número de HOST

IETF formuló varios requisitos en busca de una solución:

1. Cada **HOST** móvil debe poder usar su **DIRECCIÓN IP** principal en cualquier parte
2. No se permiten cambios de **SOFTWARE** a los **HOST FIJOS**
3. No se permiten cambios de **SOFTWARE ni a las TABLAS del ENRUTADOR**
4. La mayoría de **PAQUETES** para **HOSTS MÓVILES no debe hacer desvíos en la RUTA**
5. **No se debe incurrir en sobrecarga** cuando un **HOST MÓVIL está en casa**

La solución que se implementó fue la que se describe en la sección **5.2.9**. Cada **SITIO** que **permite vagar a sus usuarios** tiene que **crear un AGENTE DE BASE**.

Cada **SITIO** que **permite visitantes** tiene que **crear una AGENTE FORÁNEO**.

5.6.8 IPv6

Si bien **el CIDR y el NAT** pueden durar unos años más, los días de **Ipv4** están contados.

En **1990** el **IETF** comenzó a trabajar en **una nueva versión del PROTOCOLO IP**, una que **nunca de quedaría sin DIRECCIONES**, resolviera otros problemas y fuera más eficiente y flexible.

Las **metas principales fueron**

1. Manejar miles de millones de HOSTS, aún con asignación de espacio de direcciones ineficiente.
2. Reducir el tamaño de las **TABLAS DE ENRUTAMIENTO**.
3. Simplificar el **PROTOCOLO**, para permitir a los **ENRUTADORES** el procesamiento más rápido de los **PAQUETES IP**.
4. Proporcionar mayor seguridad (verificación de autenticidad y confidencialidad) que el IP actual.
5. Prestar mayor atención al **TIPO DE SERVICIO**, especialmente con datos en tiempo real.
6. Ayudar a la **multidifusión** permitiendo las especificación de alcances.
7. Posibilitar que un **HOST** sea móvil sin cambiar su dirección.
8. Permitir que el **PROTOCOLO** evolucione.
9. Permitir que el **PROTOCOLO** viejo y el nuevo coexistan por años.

IETF hizo una **convocatoria de propuestas y estudios** en el **RFC 1550**. Se recibieron **21 respuestas, no todas completas**.

En **Diciembre de 1992** se estaban tratando las **7 mejores propuestas**.

Tres de las mejores propuestas **se publicaron en 1993** en el **IEEE NETWORK** (Deering, Francis Katz y Ford).

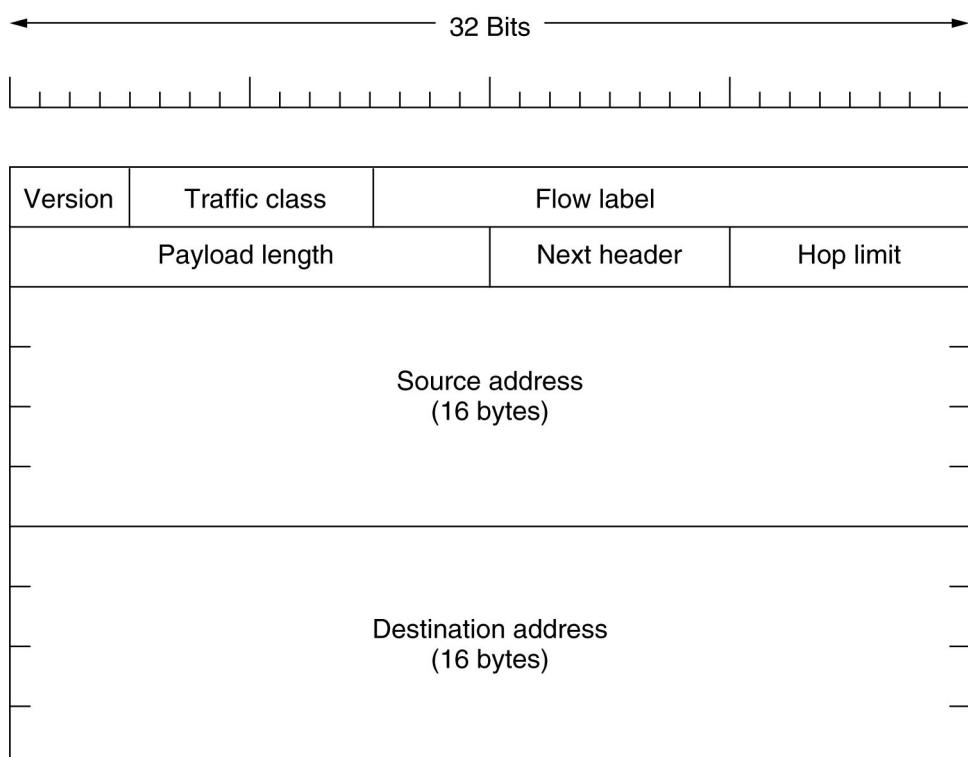
Finalmente se seleccionó una **versión modificada** de la **combinación de las respuestas de Deering y Francis**, llamada ahora **SIPP** (**PROTOCOLO SIMPLE DE INTERNET MEJORADO**) y se le dio la designación **IPv6**. Tiene las **buenas características del Ipv4, descarta las malas, y agrega nuevas donde se necesita**. En general **IPv6 ES INCOMPATIBLE con Ipv4**, pero **ES COMPATIBLE con TODOS LOS DEMÁS PROTOCOLOS Internet**, incluidos **TCP, UDP, ICMP, IGMP, OSPF, BGP y DNS**, a veces con algunas pequeñas modificaciones (principalmente para manejar direcciones más grandes). Las **principales características de IPv6** se describen en los **RFCs 2460 a 2466**.

Una dirección principal **IPv6** tiene **16 bytes (128 bits)** de longitud, que **resuelve el problema de la cantidad limitada de direcciones Ipv4** (de 4 bytes, o sea 32 bits).

El **ENCABEZADO DE UN PAQUETE IPv6** se ha simplificado y solo **contiene 7 CAMPOS**, en contraste con los **13 que tiene el ENCABEZADO DE UN PAQUETE Ipv4**. Esto permite a los **ENRUTADORES PROCESAR con mayor rapidez los PAQUETES IPv6, mejorándose la VELOCIDAD REAL DE TRANSPORTE**.

Se ha dado un **mejor apoyo a las OPCIONES, la SEGURIDAD y la CALIDAD DEL SERVICIO**.

Encabezado principal del IPv6



CAMPOS DEL ENCABEZADO DE UN PAQUETE IPv6	Función
VERSIÓN	Versión del protocolo IP
CLASE DE TRÁFICO	Distinguir entre paquetes con requisitos diferentes de entrega en tiempo real
ETIQUETA DE FLUJO	Aún es experimental. Se usará para permitir a un origen y un destino establecer una pseudoconexión con propiedades y requisitos particulares. Cada flujo está designado por la dirección de origen, dirección de destino y el valor de la etiqueta de flujo
LONGITUD DE LA CARGA ÚTIL	Indica cuántos bytes siguen al encabezado, que posee 40 bytes
ENCABEZADO SIGUIENTE	Pueden existir encabezados adicionales (opcionales) de extensión. El campo indica cuál de los seis encabezados de extensión (definidos actualmente) de haberlos, sigue a éste.
LÍMITE DE SALTOS	Se usa para evitar que los paquetes vivan eternamente. Su valor se decrementa en cada salto
DIRECCIÓN DE ORIGEN	Una dirección de 16 bytes de longitud fija. Se usa una nueva notación, escribiendo las direcciones de 16 bytes como ocho grupos de cuatro dígitos hexadecimales separados por dos puntos. Los ceros a la izquierda se pueden omitir.
DIRECCIÓN DE DESTINO	

Notar la **ausencia de los CAMPOS** que **existían** en el **ENCABEZADO DE UN PAQUETE IPv4**, como:

- **IHL:**
Pues ahora el **ENCABEZADO del PAQUETE IPv6** tiene una longitud fija.
- **PROTOCOLO:**
Porque el **CAMPO ENCABEZADO SIGUIENTE** indica lo que sigue al último **ENCABEZADO de IPv6** (ejemplo: un **SEGMENTO UDP o TCP**)
- **DESPLAZAMIENTO DEL FRAGMENTO:**
Desaparece porque **IPv6** tiene un **enfoque distinto hacia la FRAGMENTACIÓN**.
Todos los **HOSTS** que se ajustan a **IPv6** **deben DETERMINAR DINÁMICAMENTE el TAMAÑO DEL DATAGRAMA**.
El **mínimo** se incrementó de 576 a **1280 bytes, para permitir 1024 bytes de DATOS y VARIOS ENCABEZADOS**.
Cuando un **HOST envía** un **PAQUETE IPv6 demasiado grande**, en lugar de fragmentarlo, el **ENRUTADOR** que **es incapaz de reenviarlo devuelve un mensaje de error indicando al HOST que divide todos los PAQUETES futuros a ese destino**.

- **SUMA DE VERIFICACIÓN DEL ENCABEZADO:**

Desaparece, porque su **cálculo reduce en gran medida el desempeño**. Con las **redes confiables de hoy**, y además el hecho de que la **CAPA DE ENLACE DE DATOS** y la **CAPA DE TRANSPORTE ya tienen sus propias SUMAS DE VERIFICACIÓN** no se obtenía provecho al agregar otra suma de verificación.

Encabezados de extensión

Estos encabezados pueden usarse para proporcionar información extra, codificada de manera eficiente.

Hay **SEIS TIPOS** de **ENCABEZADOS DE EXTENSIÓN OPCIONALES** definidos actualmente:

ENCABEZADO DE EXTENSIÓN OPCIONAL	DESCRIPCIÓN
Opciones salto por salto	Información diversa para los enrutadores
Opciones de destino	Información adicional para el destino
Enrutamiento	Ruta total o parcial a seguir
Fragmentación	Manejo de fragmentos de datagramas
Autenticación	Verificación de la identidad del emisor
Carga útil de seguridad encriptada	Información sobre el contenido encriptado

El **ENCABEZADO DE SALTO POR SALTO** se usa para información que deben examinar **TODOS** los **ENRUTADORES A LO LARGO DE LA RUTA** y tiene el siguiente **FORMATO**:

Next header	0	194	4
Jumbo payload length			

Cuando se utiliza, el **CAMPO de LONGITUD DE LA CARGA ÚTIL** del **ENCABEZADO** del **PAQUETE IPv6** se debe establecer en **CERO**.

Este campo **ENCABEZADO DE SALTO POR SALTO** consta de:

- **ENCABEZADO SIGUIENTE**

Un **byte** que indica el **TIPO DE ENCABEZADO DE EXTENSIÓN OPCIONAL SIGUIENTE**

- **0**

Un **byte** que indica la **LONGITUD DEL ENCABEZADO DE SALTO POR SALTO** en bytes, **EXCLUYENDO LOS PRIMEROS 8 BYTES, QUE SON OBLIGATORIOS**

- **194**

• **4**

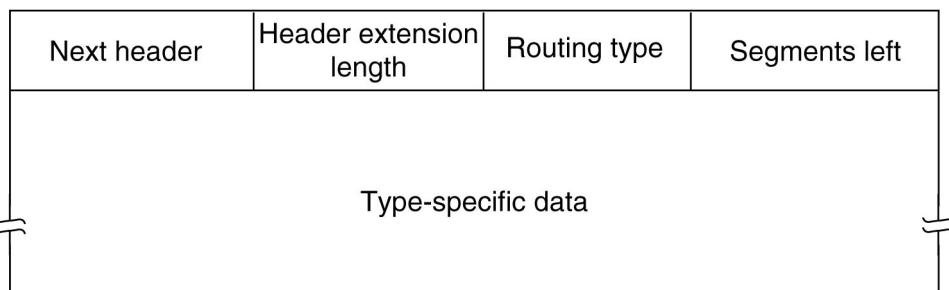
Código que define el tamaño del DATAGRAMA como número de 4 bytes

- **LONGITUD DE LA CARGA ÚTIL GRANDE**

Los **últimos 4 bytes (obligatorios)**, indican el **tamaño del DATAGRAMA**. No se permiten los tamaños menores que **65.536**, pues **causará que el ENRUTADOR descarte el paquete y devuelva un mensaje ICMP de error**. Los **DATAGRAMAS** que usan este **ENCABEZADO DE EXTENSIÓN** se llaman **JUMBOGRAMAS**, estos son **importantes para aplicaciones de SUPERCOMPUTADORAS** que deben **TRANSFERIR con EFICIENCIA GIGABYTES DE DATOS** a través de **INTERNET**.

El **ENCABEZADO DE OPCIONES POR DESTINO** está proyectado para **CAMPOS** que sólo necesitan **SER INTERPRETADOS** en el **HOST de DESTINO**

El **ENCABEZADO DE ENRUTAMIENTO** lista **uno o más ENRUTADORES** que **deben ser visitados** en el **caminio al DESTINO**. Tiene el siguiente formato:



Sus **primeros 4 bytes** contienen **cuatro enteros de un byte**:

- **ENCABEZADO SIGUIENTE**
- **LONGITUD DEL ENCABEZADO DE EXTENSIÓN**
- **TIPO DE ENRUTAMIENTO**
Da el formato al resto del encabezado:

- **CERO**

Significa que una palabra de **32 bits sigue a la primera palabra**, seguida de **algún número de DIRECCIONES IPv6**

- Pueden inventarse otros valores en el futuro

- **SEGMENTOS DE LA IZQUERDA**

Registra **cuántas DIRECCIONES IPv6 de la LISTA no se han visitado todavía**.

Se reduce cada vez que se visita una de esas DIRECCIONES IPv6.

Cuando está en cero, en general está **tan cerca del DESTINO que la mejor RUTA es obvia**.

El **ENCABEZADO DE FRAGMENTO** maneja la **FRAGMENTACIÓN** de una manera **similar** a como lo hace **IPv4**.

El **ENCABEZADO DE AUTENTICACIÓN** proporciona un mecanismo mediante el cuál el **RECEPTOR** de un **PAQUETE IPv6** puede estar seguro de **quién lo envió**. La **CARGA ÚTIL DE SEGURIDAD ENCRIPCIÓNADA** posibilita **ENCRIPRAR** el **CONTENIDO** de un **PAQUETE IPv6** de modo que sólo el **RECEPTOR PRETENDIDO** pueda leerlo. **Se usan técnicas criptográficas para lograrlo.**

Controversias

Muchas de las decisiones tomadas para el IPv6 son temas de fuertes controversias:

- la **longitud** de las **DIRECCIONES IPv6**
- la **longitud** del **CAMPO LÍMITE DE SAÍDOS**
- la **desaparición del CAMPO** de **SUMA DE VERIFICACIÓN**
- los **HOSTS MÓVILES** que generan **asimetrías en el ENRUTAMIENTO** conforme **se mueven de un lugar, al otro lado del mundo**. No se pudo generar consenso en ninguna propuesta
- la **SEGURIDAD**, un **aspecto fundamental**