IP Routing

- Routing table:

IP addr - prefix	Mask		Gateway	Metric	Interface	Flags
45.176.90.242	255.255.255.255	32		1	ppp0	
192.168.1.0	255.255.255.0	24		1	eth0	
192.168.2.0	255.255.255.0	24		1	eth1	
172.16.0.0	255.255.0.0	16	192.168.1.254	1	eth0	
0.0.0.0	0.0.0.0	0		1	ppp0	

- Ordenada por:
  1. Máscaras: de mascaras más grandes a más chicas
  2. Métrica: de más chica a más grande
  3. Otros criterios según el OS

IP Routing

IP addr - prefix	Mask		Gateway	Metric	Interface	Flags
45.176.90.242	255.255.255.255	32		1	ppp0	
192.168.1.0	255.255.255.0	24		1	eth0	
192.168.2.0	255.255.255.0	24		1	eth1	
172.16.0.0	255.255.0.0	16	192.168.1.254	1	eth0	
0.0.0.0	0.0.0.0	0		1	ppp0	

- Se recorre la tabla ordenada, se aplica (operación bitwise AND) la máscara a la dirección de destino y si coincide con el prefix, se envía al gateway por la interface.

IP Routing

- Cuando se agrega una dirección de IP/máscara a una interface, se genera automáticamente una entrada en la routing table para poder acceder a los dispositivos directamente conectados.

- Ruta por default, prefix 0.0.0.0, máscara 0.0.0.0. Match all. Posibilidad de multiples rutas default con distintas métricas.

IP Routing

- Linux
  - **ip route**
  - **netstat -rn**
  - **route**

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         r               0.0.0.0         UG    100    0        0 ens33
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
192.168.8.0     0.0.0.0         255.255.255.0   U     100    0        0 ens33
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```

```
default dev ppp1 scope link
45.176.90.242 dev ppp1 proto kernel scope link src 45.176.90.29
192.168.0.0/24 dev enp4s0 proto kernel scope link src 192.168.0.254
192.168.2.0/24 dev enp4s1 proto kernel scope link src 192.168.2.254
192.168.8.0/24 dev enp3s0 proto kernel scope link src 192.168.8.254
200.51.241.1 dev ppp0 proto kernel scope link src 191.80.209.86
```

IP Routing

- Windows:
  - **`netstat -rn`**
  - **`route print`**

```
IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0    192.168.8.254      192.168.8.4     281
        127.0.0.0        255.0.0.0         On-link        127.0.0.1     331
        127.0.0.1  255.255.255.255         On-link        127.0.0.1     331
  127.255.255.255  255.255.255.255         On-link        127.0.0.1     331
      192.168.8.0    255.255.255.0         On-link      192.168.8.4     281
      192.168.8.4  255.255.255.255         On-link      192.168.8.4     281
    192.168.8.255  255.255.255.255         On-link      192.168.8.4     281
     192.168.42.0    255.255.255.0         On-link     192.168.42.1     291
     192.168.42.1  255.255.255.255         On-link     192.168.42.1     291
   192.168.42.255  255.255.255.255         On-link     192.168.42.1     291
     192.168.48.0    255.255.255.0         On-link     192.168.48.1     291
     192.168.48.1  255.255.255.255         On-link     192.168.48.1     291
   192.168.48.255  255.255.255.255         On-link     192.168.48.1     291
        224.0.0.0        240.0.0.0         On-link        127.0.0.1     331
        224.0.0.0        240.0.0.0         On-link     192.168.42.1     291
        224.0.0.0        240.0.0.0         On-link     192.168.48.1     291
        224.0.0.0        240.0.0.0         On-link      192.168.8.4     281
  255.255.255.255  255.255.255.255         On-link        127.0.0.1     331
  255.255.255.255  255.255.255.255         On-link     192.168.42.1     291
  255.255.255.255  255.255.255.255         On-link     192.168.48.1     291
  255.255.255.255  255.255.255.255         On-link      192.168.8.4     281
===========================================================================
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
          0.0.0.0          0.0.0.0    192.168.8.254  Default
===========================================================================
```

IP Routing

- El paquete IP tiene direcciones de origen y destino.
- Cómo se envía un paquete a un gateway? Problema de la capa 2.
- En redes de multiple acces, para enviarlo al gateway se arma un frame con el gateway como destino, utilizando MAC address del GW.

Routing Algorithms

- Software responsable de decidir por que línea o interface un paquete debe ser transmitido.
  - Datagram: para cada paquete
  - VC: única vez en el setup de la conexión. Session routing.

- Forwarding: decidir por que interface se va a enviar un paquete utilizando las tablas de routing.

- Llenado y actualización de routing tables: responsabilidad del algoritmo de routeo.

Routing Algorithms

Propiedades:

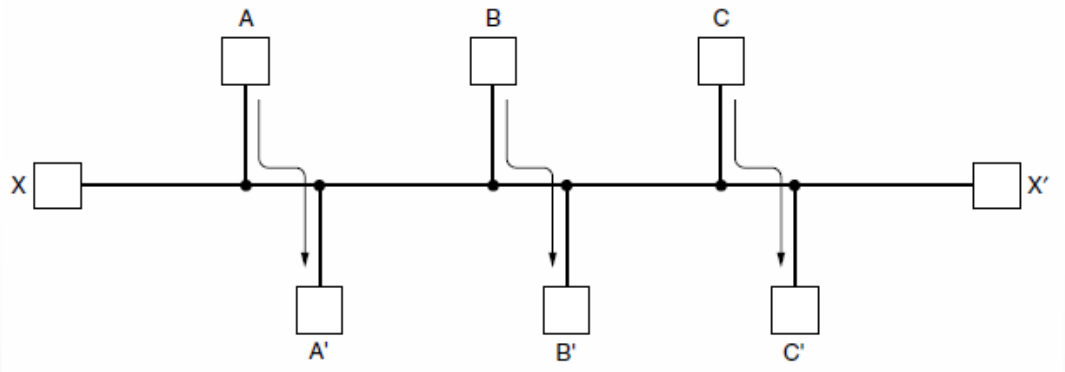
- Correctos y simples
- Robustos
  - Soportar cambios de topología por caídas de líneas/routers.
- Estabilidad
  - Convergencia a un conjunto de caminos.
  - Rapidez de convergencia.
- Justos
- Eficientes
  - Competencia entre justos y eficientes

Routing Algorithms

Optimización por:

- Delay medio de paquetes
- Troughput

Competencia entre algoritmos justos y eficientes



Routing Algorithms

- Algoritmos no adaptativos. Static routing.
  - Se precalcula todas las rutas para llegar de *I* a *J*, para todo *I*, *J*.
  - No responde a fallas.

- Algoritmos adapatativos. Dynamic routing.
  - Cambian sus decisiones de ruteo reflejando los cambios en la topología de la red (o del tráfico).
  - Varían según:
    - Donde sacan información: localmente, routers adyacentes, todos los routers.
    - Cuando son actualizadas las rutas: ante cambios de topología o periódicamente.
    - Métrica utilizada para optimización: distancia, número de hops, tiempo estimado de tránsito.

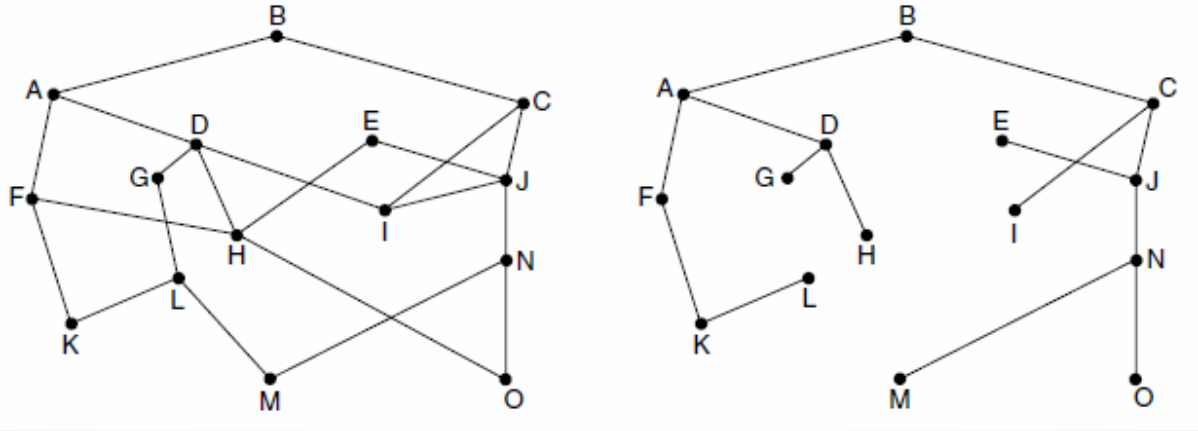
Routing Algorithms

- Principio de optimalidad

Si un router *J* está en el camino óptimo entre el router *I* y el router *K*, entonces el camino óptimo entre *J* y *K* va por la misma ruta.

Consecuencia: El conjunto de rutas óptimas a un destino es un árbol con raíz en el destino. Sink tree. Puede no ser único. Sin loops.

Sink Tree para B:



Shortest Path

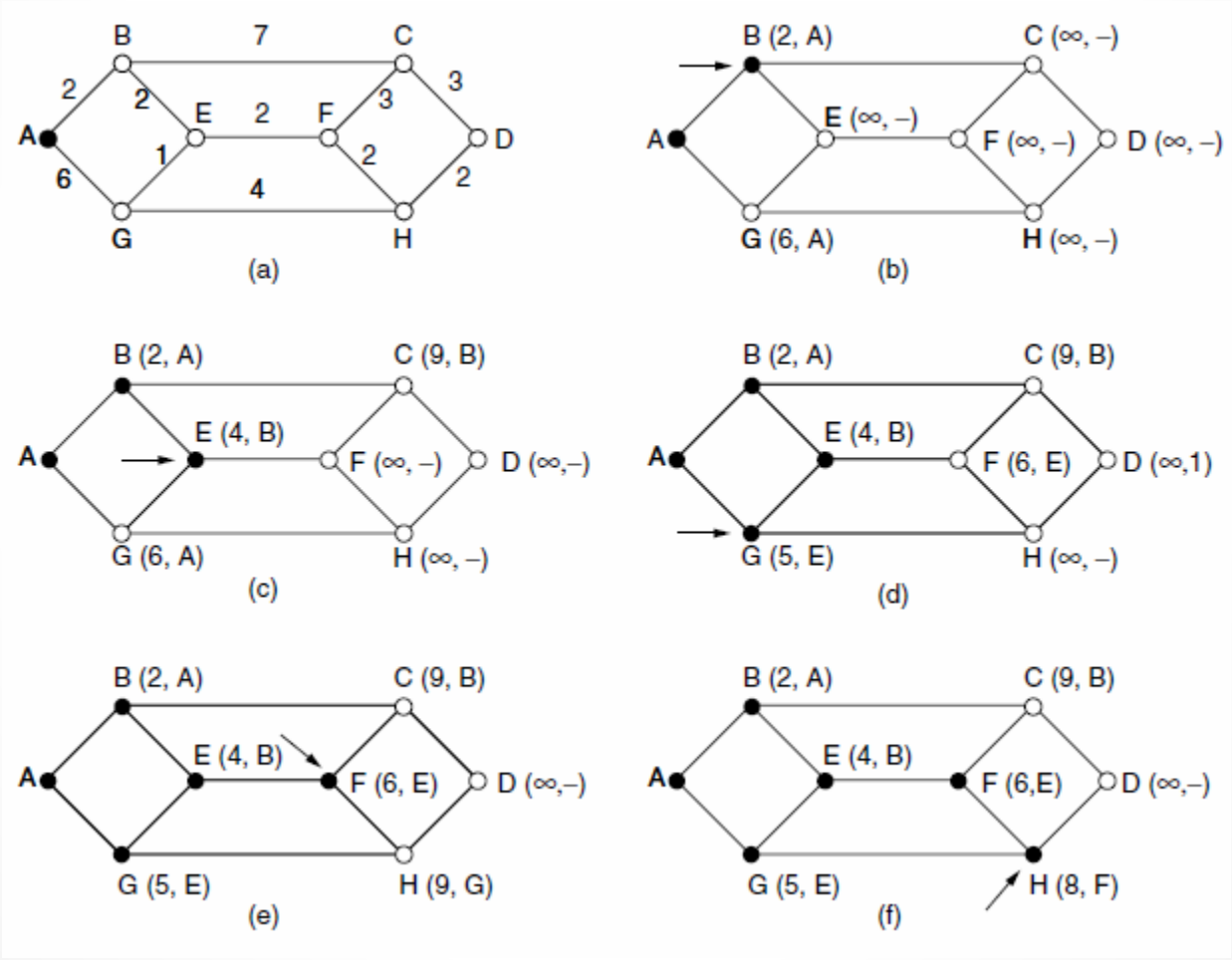
Se construye un grafo de la red con los router como nodos y se encuentra el camino más corto entre nodos.

Métricas: número de hops, distancia geográfica, tiempo medio de delay medido periódicamente, bandwith, tráfico, costo, etc.

Cada nodo tiene un label con la distancia al nodo de origen a través del mejor camino conocido.

Dijkstra, 1959

Shortest Path



Flooding

Los routers toman decisiones de acuerdo a su conocimiento local, no al mapa completo de la red.

- Enviar el paquete por todas las líneas excepto por la que llegó.

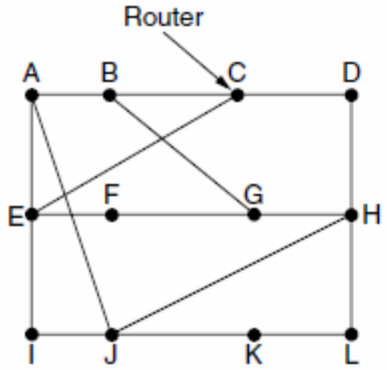
- Genera muchos paquetes repetidos, infinitos si no se limita el tiempo de vida.
- Se limitan retransmisiones del mismo paquete poniendo un número de secuencia en el paquete y evitando reenviar un paquete ya enviado. Cada router guarda el número de seq. más grande visto del router de origen.
- No es práctico, muy ineficiente. Útil para broadcasts. Robusto. Poco setup. Siempre elige el camino más corto, prueba en paralelo. Delay más corto. Benchmark de otros algoritmos.

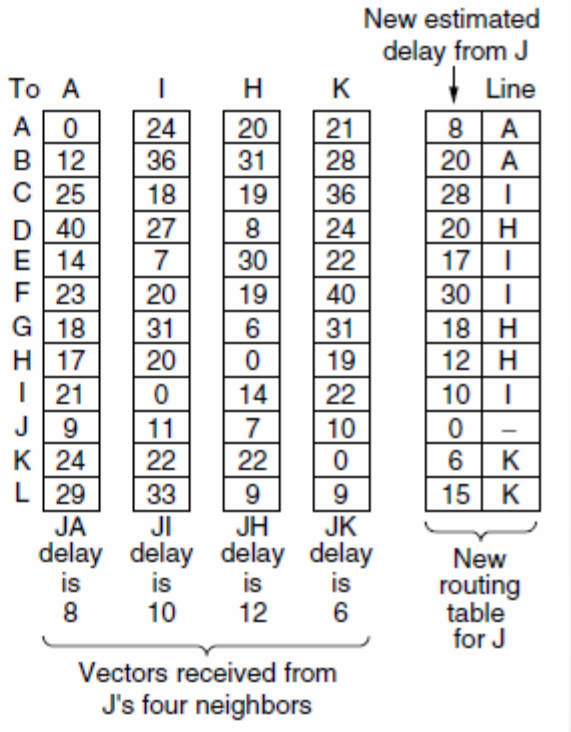
Distance Vector Routing

- Cada router mantiene una tabla (vector) de con la mejor distancia conocida a cada destino y que link utilizar. Las tablas se actualizan intercambiando información entre routers vecinos.

Bellman-Ford

RIP

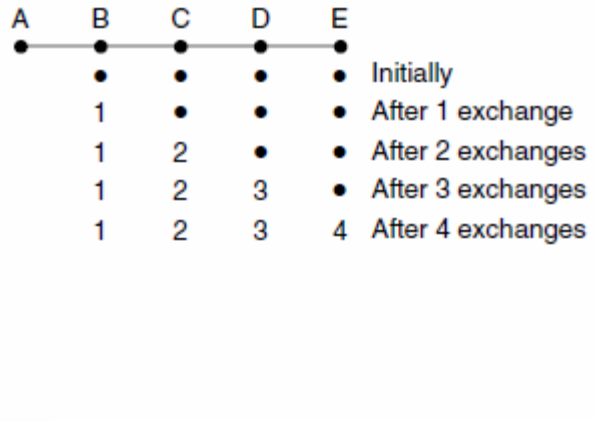


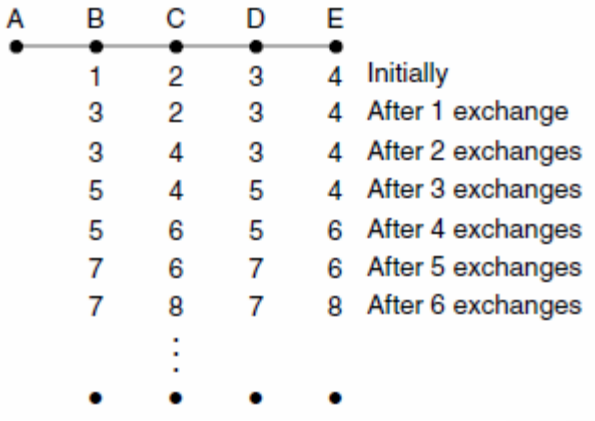


Distance Vector Routing

Converge al resultado correcto, pero es muy lento.

Reacciona rápido a buenas noticias y muy mal a las malas.





Count-to-Infinity problem.

Se tarda infinito en dar de baja un router, se setea infinito en N+1.

Un router no sabe si está en el path que le informan.

Link State Routing

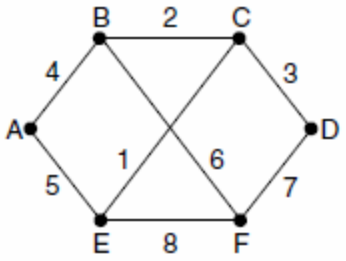
1. Cada router descubre sus vecinos y aprende sus direcciones.
2. Setea la distancia o métrica a cada uno de sus vecinos.
3. Construye y envía un paquete con lo que sabe.
4. Recibe paquetes con el conocimiento de todos los otros routers en la red.
5. Computa el shortest path a cada otro router, con el algoritmo de Dijkstra.

IS-IS

OSPF

Link State Routing

Link State Packets: ID, número de secuencia y edad.



Link State Packets

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

Se distribuyen por flooding. Para evitar paquetes repetidos se utiliza el número de secuencia que incrementa el que los envía.

Se mantiene una lista de (router, número de secuencia).

Problemas: overflow de número de secuencia → 32 bits

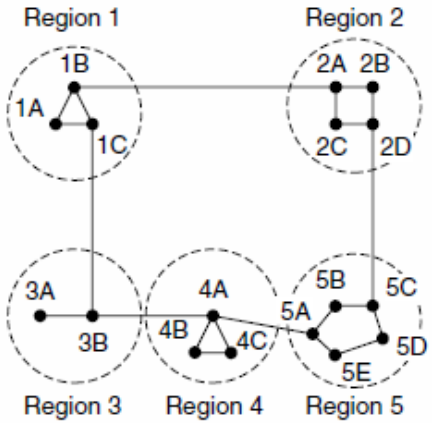
Errores de número de secuencia. Reinicio de un router.

Se agrega la edad que se va disminuyendo una vez por segundo, cuando llega a 0 se descarta la información.

Hierarchical Routing

Cuando la red crece se hace imposible que cada router sepa de todo otro router. La solución es hacer routing jerárquico.

La red se divide en regiones y cada router sólo conoce la topología dentro de su región.



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Hierarchical Routing

¿Cuantos niveles debería tener una jerarquía?

Ejemplo de 720 routers:

1. 720 entradas.
2. 24 regiones de 30 routers: 30 + 23 = 53 entradas.
3. 8 clusters de 9 regiones de 10 routers: 10 + 8 + 7 = 25 entradas.

Óptimo para *N* routers es **ln** *N* niveles, requiriendo *e* **ln** *N* entradas por router.

Broadcast Routing

❖ Enviar un paquete a todos los destinos posibles.

- Se puede enviar un paquete distinto a cada destino. Desperdicio de ancho de banda y conocimiento de todos los destinos. No se necesita nada especial.

- Multidestination routing: se envía un paquete con la lista de todos los destinos. Se va modificando la lista a medida que pasa por los routers. Se ahorra ancho de banda, pero hace falta conocer todos los destinos. Bastante trabajo para los routers.

Broadcast Routing

- Flooding con número de secuencia para evitar duplicados.

- Reverse path forwarding: se verifica si el paquete llegó desde la mejor ruta hacia el origen del brodcast. Si es así, se reenvía por los otros links. Si no, debe ser un duplicado y se descarta.