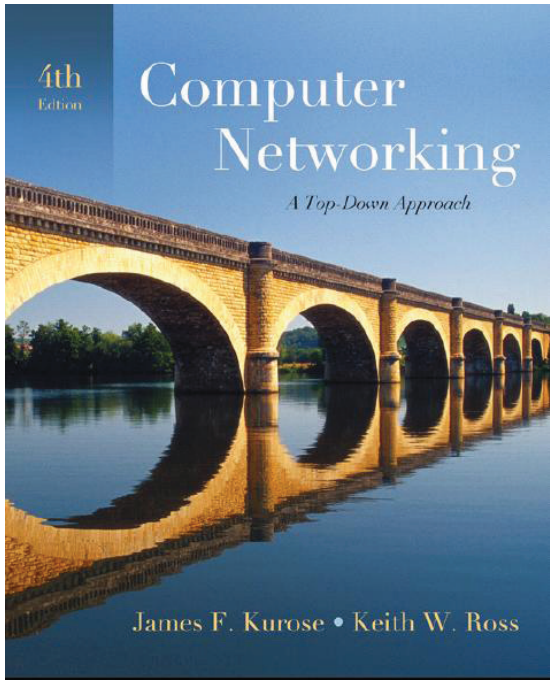
Introducción a las Redes de Computadoras

Capítulo 4
Capa de Red



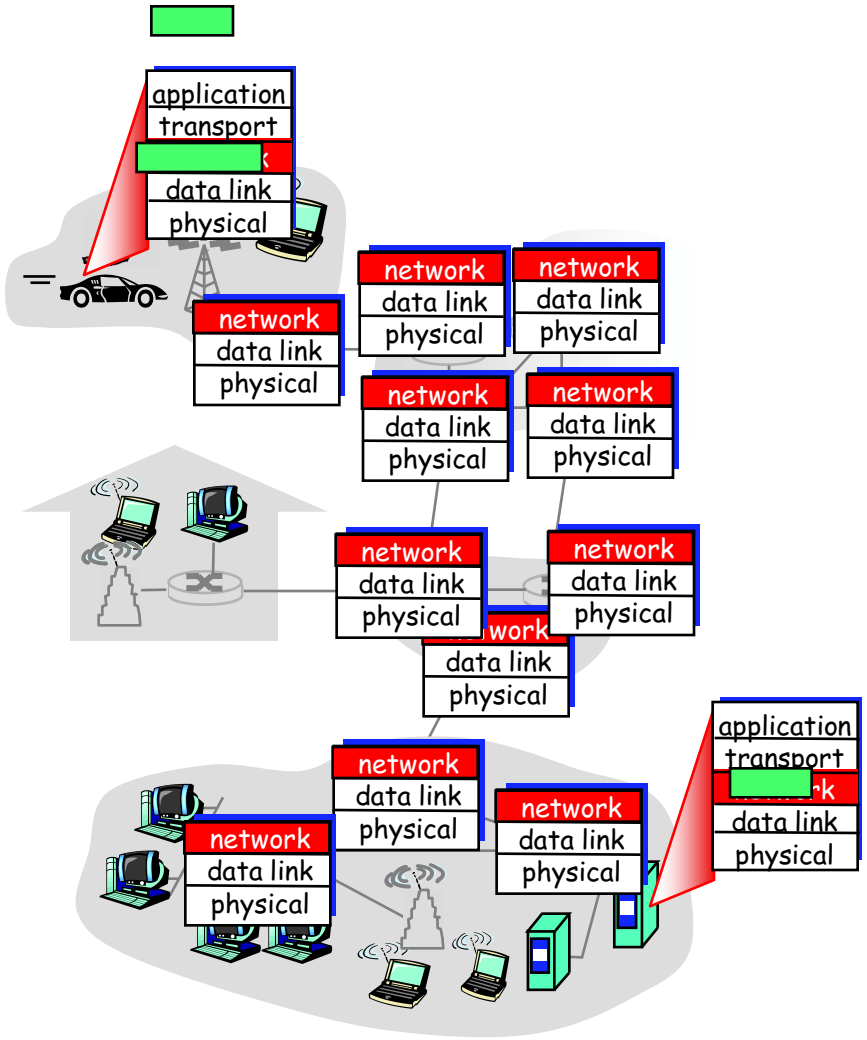
Computer Networking: A Top Down Approach, 4th edition.
Jim Kurose, Keith Ross
Addison-Wesley, July 2007.

Nota acerca de las transparencias del curso:

Estas transparencias están basadas en el sitio web que acompaña el libro, y han sido modificadas por los docentes del curso.

Capa de Red

- transporte de segmentos desde host emisor a receptor
- el emisor encapsula segmentos en datagramas
- el receptor entrega segmentos a la capa de transporte
- Protocolos de capa de red *deben* estar presentes en cada host y router
- Los routers examinan el cabezal de todos los datagramas IP que reciben



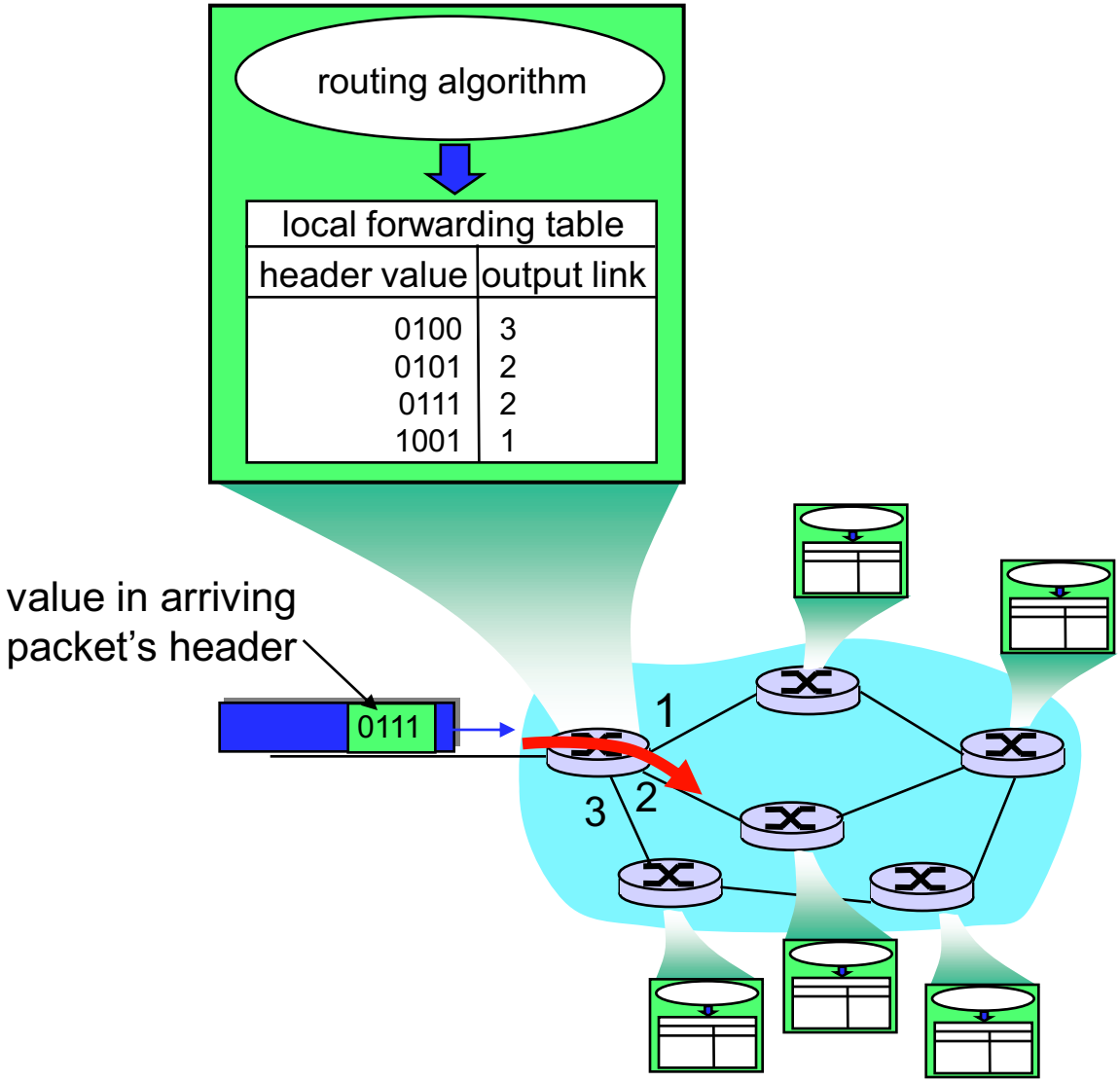
Dos funciones clave de la Capa de Red

- *forwarding:* mover paquetes entre puertos de entrada y salida del router
- *enrutamiento:* determinar la ruta de los paquetes desde origen a destino
  - *algoritmos de enrutamiento*

analogía:

- *enrutamiento* : proceso de planificación del viaje (p. ej. en avión) desde la salida a la llegada
- forwarding: proceso de pasar por cada punto de intercambio (p. ej. cambio de puertas en aeropuerto)

Interacción entre routing & forwarding



Establecimiento de la conexión

- 3a funcionalidad importante en *algunas* tecnologías de red:
  - ATM, frame relay, X.25
- antes de iniciar el flujo de datagramas, los extremos (hosts) y los routers intervinientes deben establecer una conexión virtual
  - routers están implicados
- servicio de conexión de capa de red vs. capa de transporte:
  - red: entre hosts (involucra routers cuando se establecen VCs)
  - transporte: entre dos procesos

Servicios de Capa de Red orientados y no-orientados a conexión

- datagramas: servicio no-orientado a conexión
- circuitos virtuales (VC): servicio orientado a conexión
- análogo a los servicios de capa de transporte, pero:
  - servicio: host-a-host
  - no se puede elegir: la red provee uno u el otro
  - implementación: en el "core" de la red

Circuitos Virtuales

"el camino de extremo a extremo se comporta como un circuito telfónico"

- tiene en cuenta parámetros de performance
- la red es responsable a lo largo del camino

- establecimiento de llamada antes del flujo de datos
- cada paquete tiene un identificador de VC (no la dirección del host de destino)
- *cada* router en el camino mantiene el estado de cada conexión
- se pueden *asignar* recursos de routers y enlaces (ancho de banda, buffers) para cada VC (recursos dedicados = servicio predecible)

Implementación de VCs

un VC consiste de:

1. camino (path) de fuente a destino
2. número de VC, uno por cada enlace a lo largo del camino
3. entradas en las tablas de forwarding de los routers a lo largo del camino: cross-conexiones

- los paquetes de un VC usan el número que lo identifica (en lugar de la dirección de destino)
- el número del VC puede cambiar en cada enlace.
  - los números de VC tienen alcance local
  - el nuevo número de VC sale de la tabla de forwarding

Tabla de forwarding

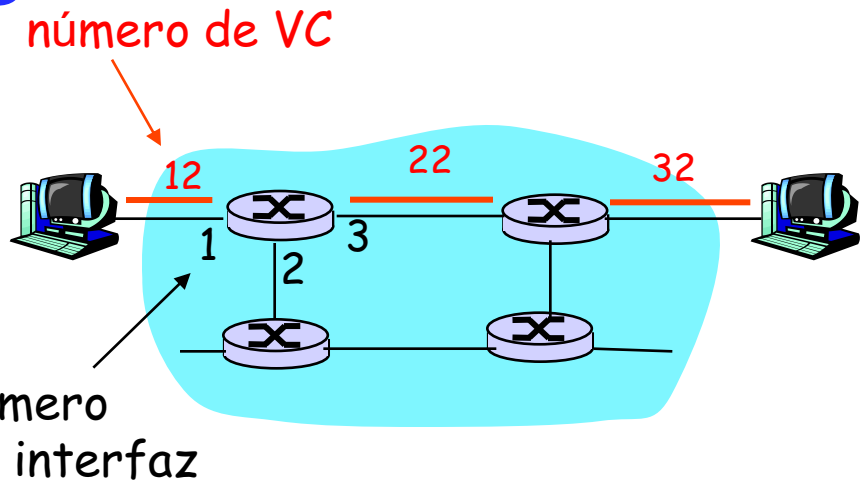


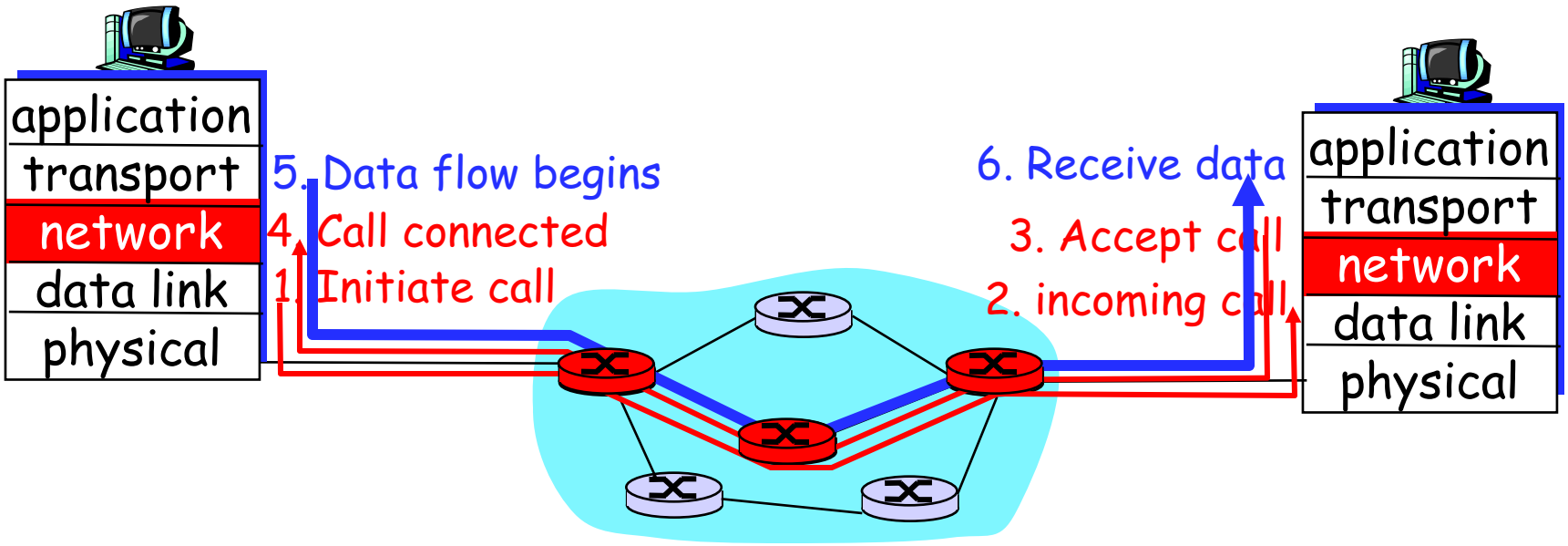
Tabla de forwarding en el router "nor-oeste":

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

Los routers mantienen información de estado de las conexiones!

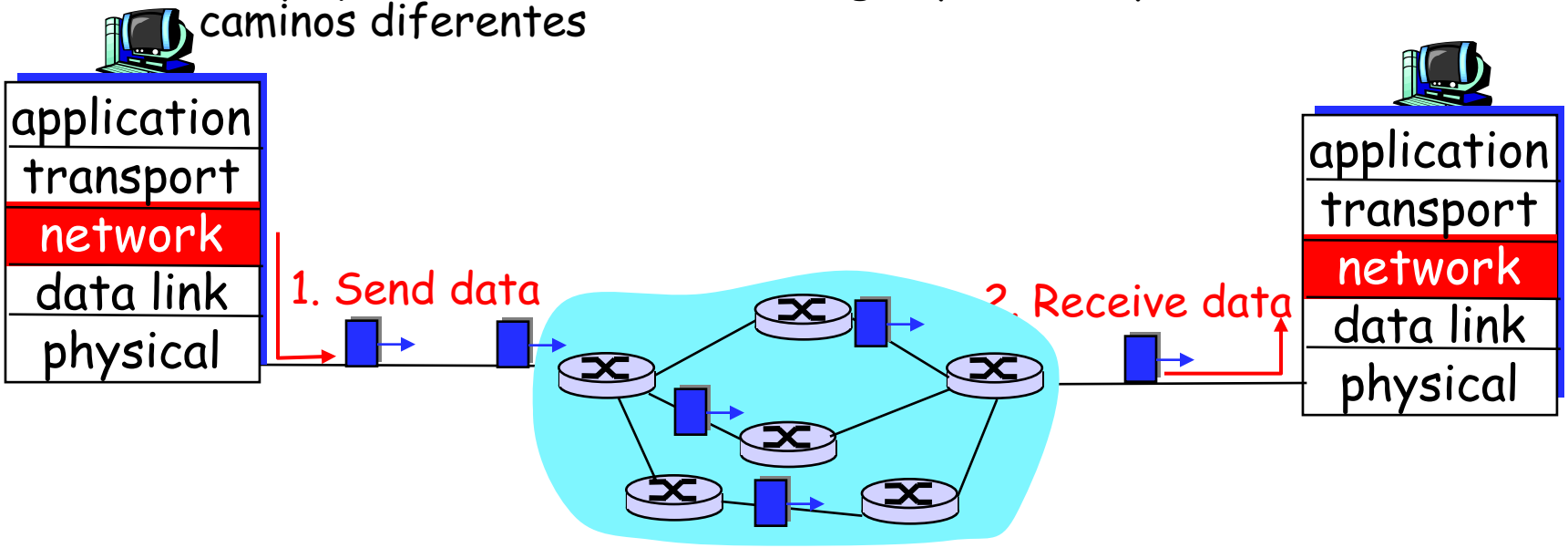
Circuitos Virtuales: protocolos de señalización

- Usados para establecer, mantener y dar de baja los VCs
- usados en ATM, frame-relay, X.25
- "no se utilizan en Internet"
  - RSVP, MPLS…



Redes de datagramas

- no existe procedimeinto de establecimiento de conexión en capa de red
- routers: no mantienen estado de conexiones extremo a extremo
  - no existe en concepto de "conexión" a nivel de red
- los paquetes son encaminados utilizando la dirección de host destino
  - Los paquetes entre un mismo origen y destino pueden tomar caminos diferentes



Datagramas o VCs: por qué?

Internet (datagramas)

- intercambio de datos entre computadores
  - servicio "elástico", sin requerimientos estrictos de tiempo
- end systems "inteligentes" (computadores)
  - adaptables, control de flujo y recuperación ante errores
  - red simple, "borde" complejo
- muchos tipos de enlaces
  - características diferentes
  - servicio uniforme

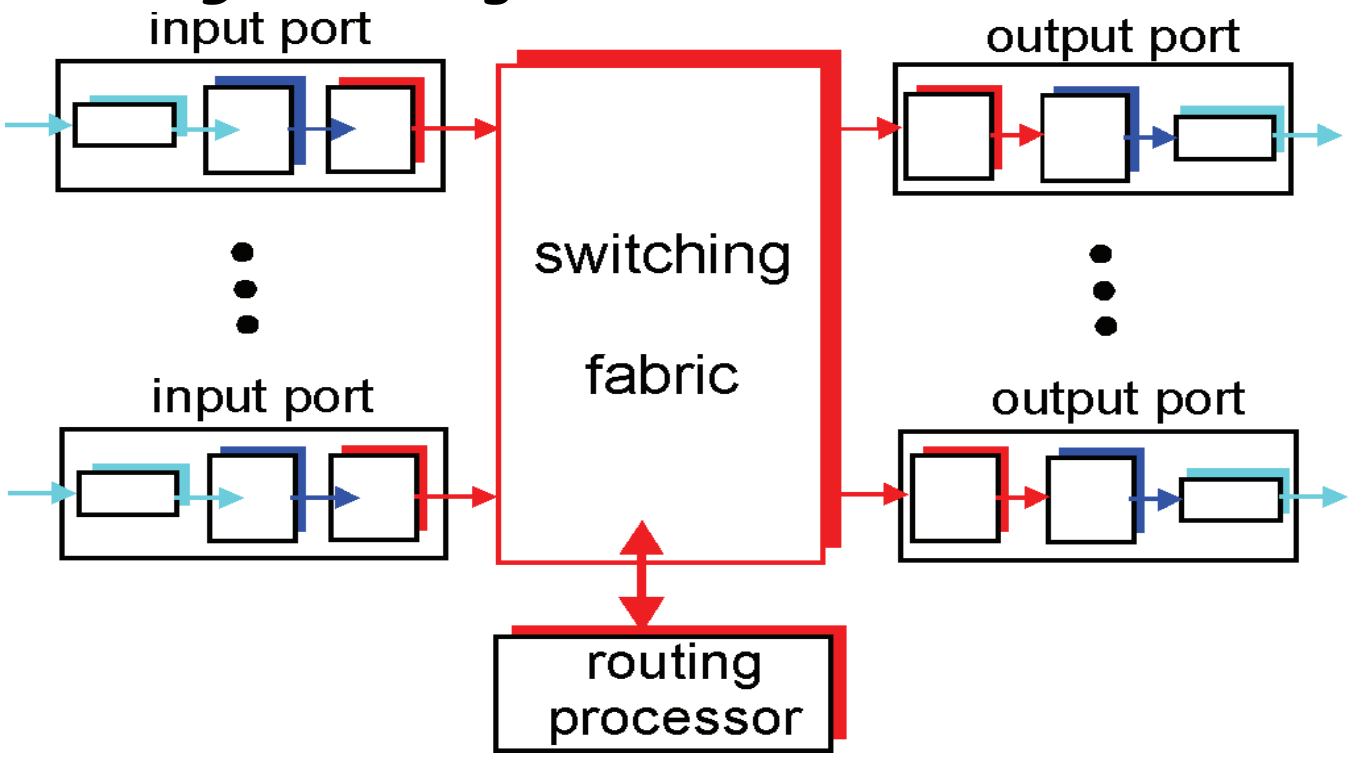
ATM (VC)

- evolución desde la telefonía
- conversación entre seres humanos:
  - reqs. de tiempo estrictos, se necesita un servicio confiable
  - y garantizado
- end systems "tontos"
  - teléfonos
  - complejidad *en* la red

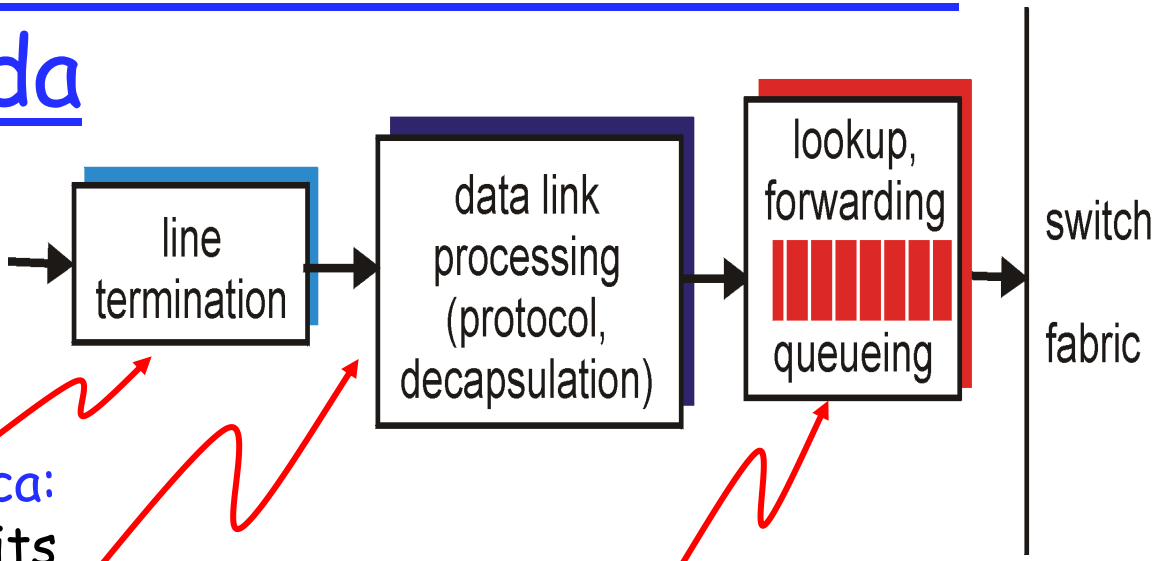
Arquitectura del Router

Dos funciones fundamentales:

- ejecutar los algoritmos/protocolos de routing (RIP, OSPF, BGP)
- *forwarding de* datagramas entre entrada/salida



Funciones de los Puertos de Entrada



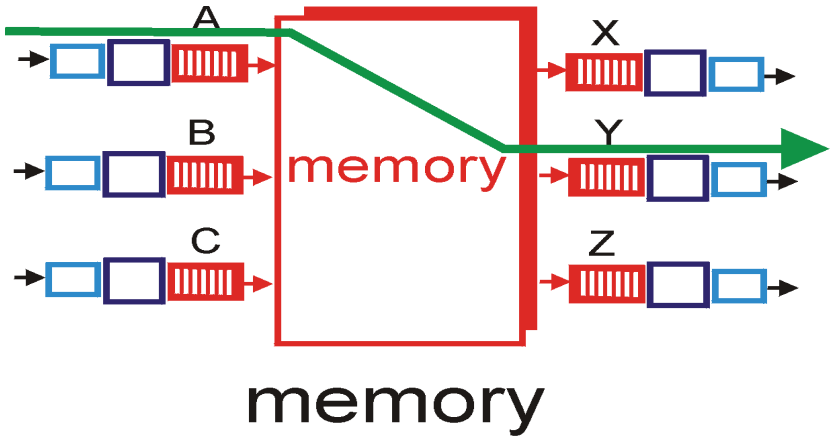
Capa física:
recepción de bits

Capa de enlace:
ej., ethernet
ver cap. 5

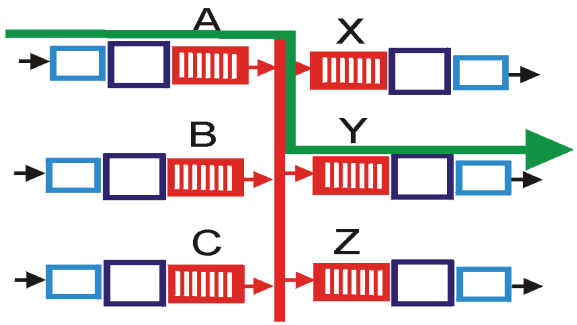
Conmutación descentralizada:

- Dado el destino del datagrama, buscar puerto de salida usando tabla de forwarding en memoria del puerto de entrada
- objetivo: completar procesamiento de entrada a 'velocidad de línea'
- queuing (encolamiento): si los datagramas llegan más rápido que la tasa de forwarding hacia el "switch fabric"

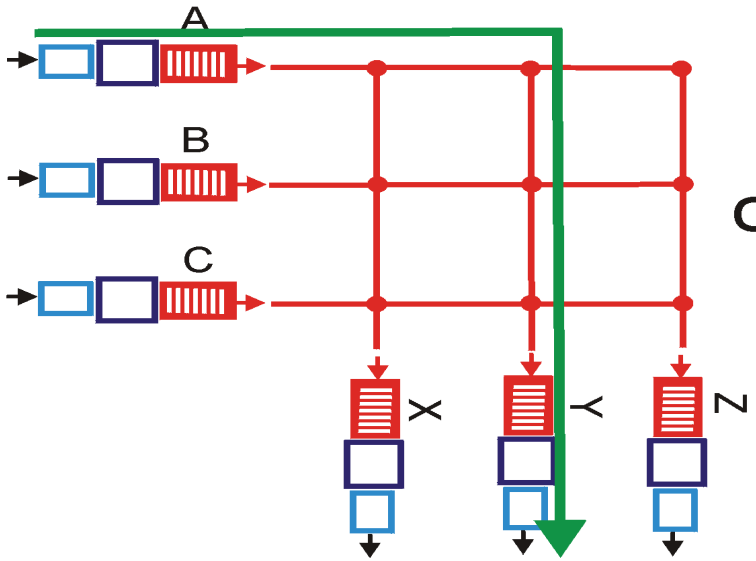
Tipos de switching fabrics (matrices de conmutación)



memory



bus

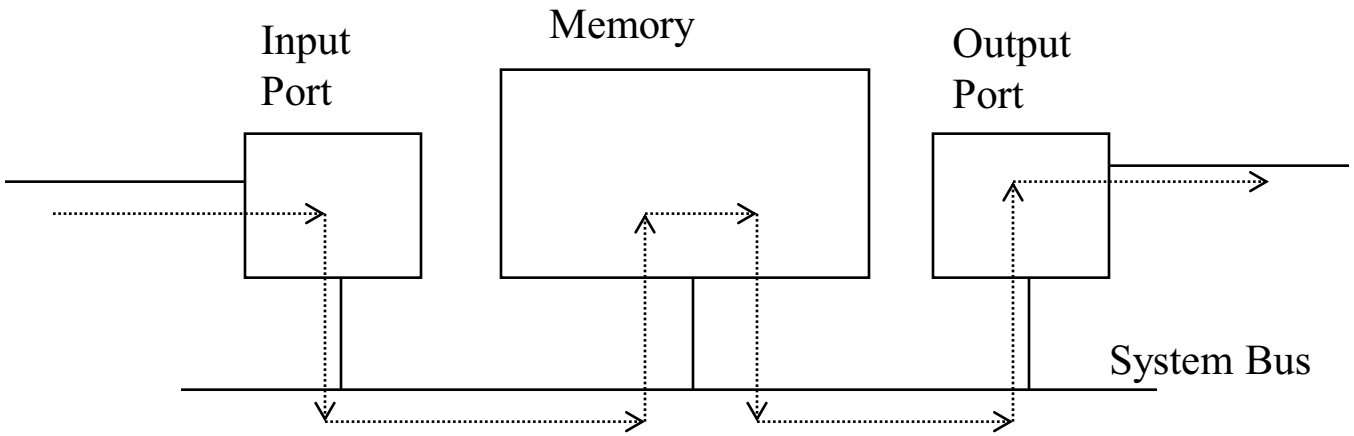


crossbar

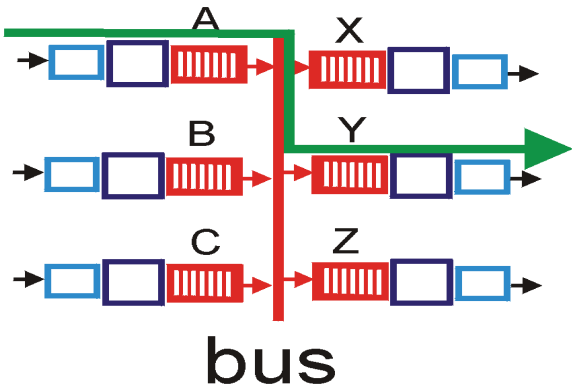
Conmutación en memoria

Routers de primera generación :

- computador tradicional con conmutación controlada directamente por la CPU
- los paquetes se copian a la memoria del sistema
- velocidad limitada por el ancho de banda de memoria (2 accesos al bus por datagrama)



Conmutación en el bus

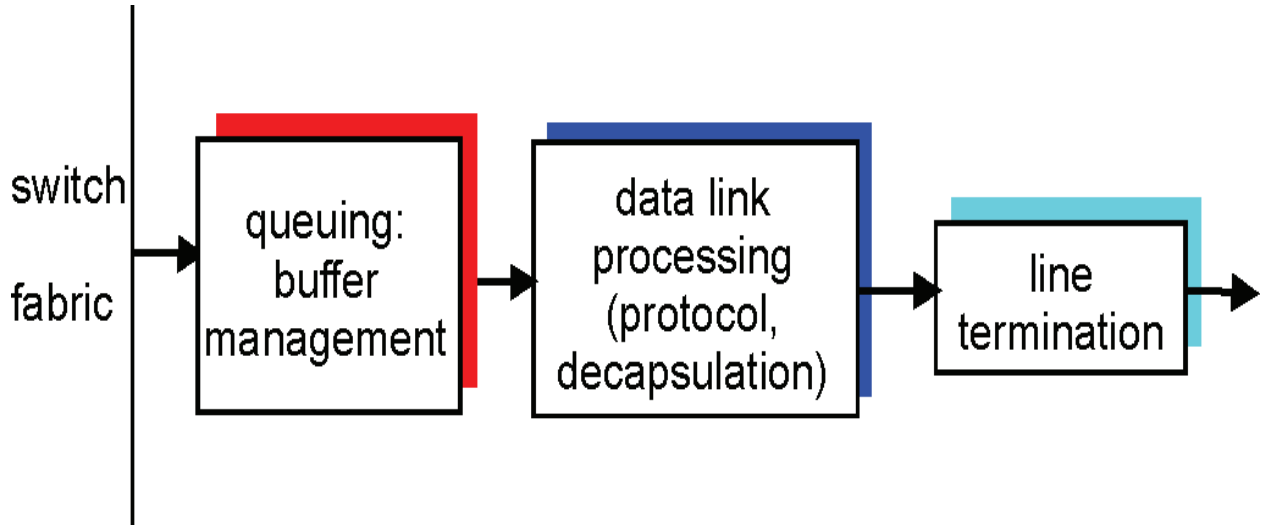


- el datagrama se copia del puerto de entrada al de salida por el bus compartido
- bus contention: velocidad de conmutación limitada por el anchio de banda del bus
- Ejemplo: 32 Gbps bus, Cisco 5600: velocidad suficiente para routers de acceso y empresariales

Conmutación con una red de interconexión

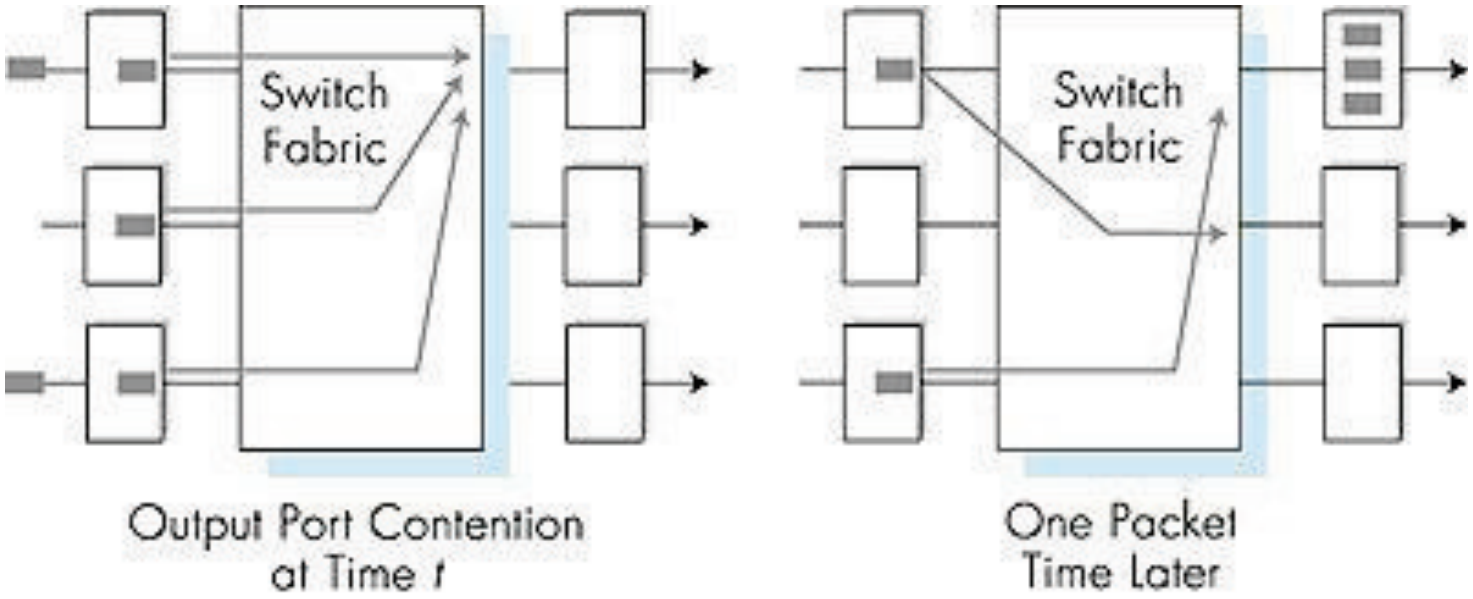
- supera las limitaciones de ancho de banda del bus
- redes de Banyan y otras inicialmente desarrolladas para interconexión de sistemas multiprocesadores
- diseño avanzado: fragmentación de datagramas en celdas de tamaño fijo, que se conmutan en la matriz
- Ej. Cisco 12000: conmuta 60 Gbps a través de la red de interconecxión

Puertos de Salida



- *buffering* (almacenamiento) requerido cuando los datagramas llegan desde la matriz más rápido que la tasa de trasmisión
- *disciplina de scheduling* (despacho) elige dtagramas en la cola para ser trasmitidos

Colas en Puertos de Salida



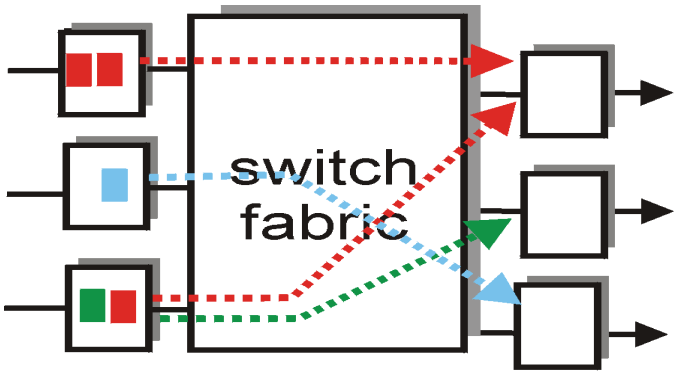
- *buffering* (almacenamiento) requerido cuando los datagramas llegan desde la matriz más rápido que la tasa de trasmisión
- *pueden existir retardos y pérdidas debido a overflow del buffer del puerto de salida!*

Tamaño del buffer?

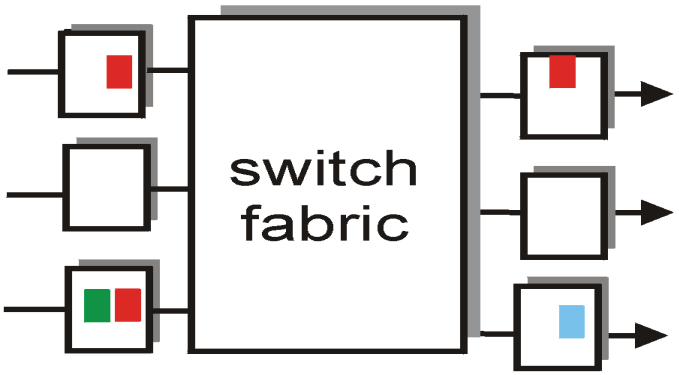
- regla de uso, RFC 3439: "buffering" promedio igual al producto del RTT típico (digamos 250 mseg) por la capacidad del enlace *C*
  - ej., *C* = 10 Gps: buffer=2.5 Gbit
  - asume relativamente pocos flujos TCP
- Recomendaciones recientes: con *N* flujos, el buffer debe ser $\frac{RTT \cdot C}{\sqrt{N}}$
  - asume N grande

Colas en Puerto de Entrada

- si la matriz es más lenta que la combinación de los puertos de entrada -> se produce encolamiento
- bloqueo Head-of-the-Line (HOL): datagrama encolado al frente impide progresar al resto de la cola
- *pueden existir retardos y pérdidas debido a overflow del buffer del puerto de entrada!*



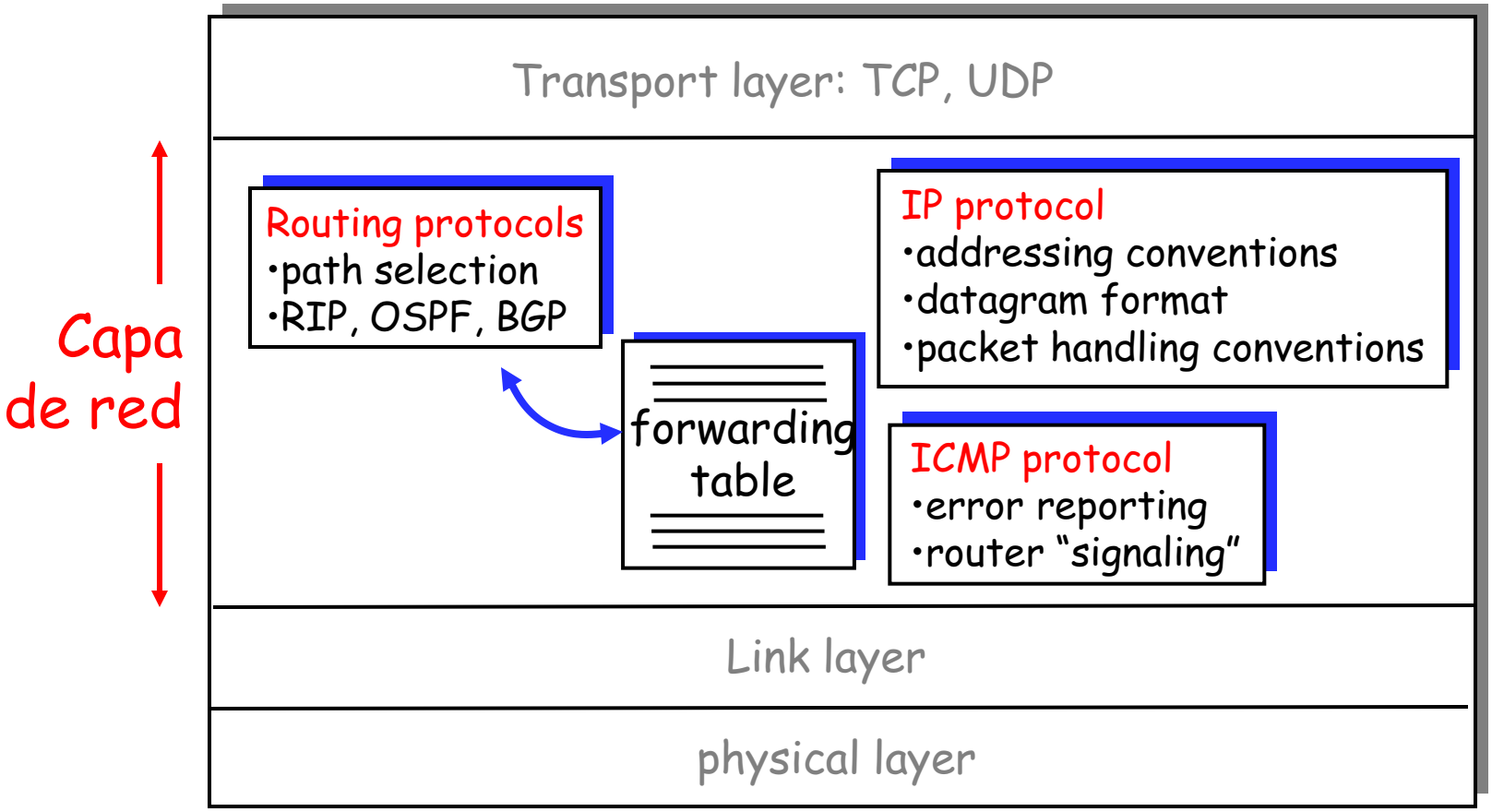
output port contention
at time t - only one red
packet can be transferred



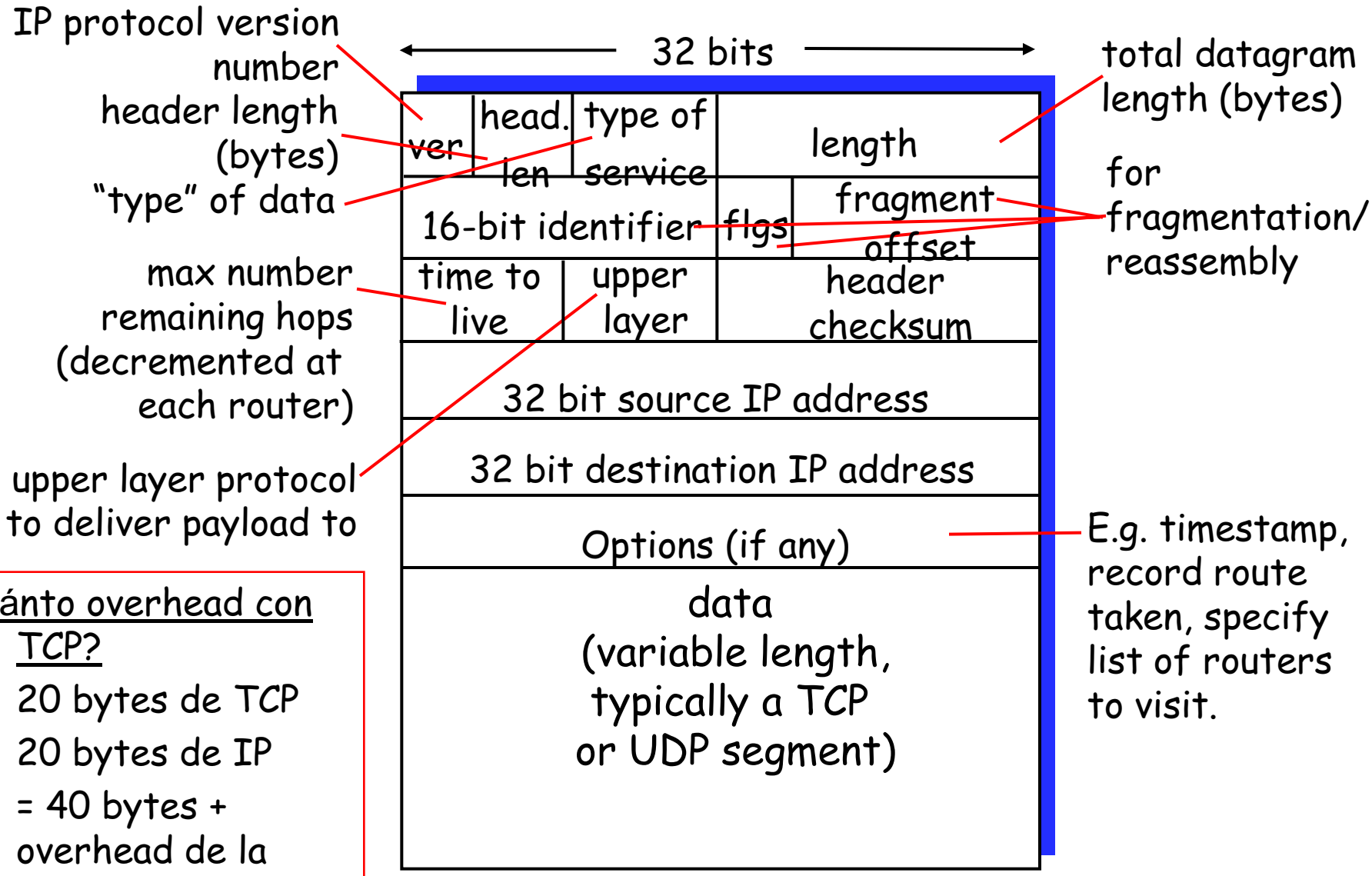
green packet
experiences HOL blocking

La Capa de Red en Internet

Funciones de capa de red en hosts y routers:



Formato del datagrama IP

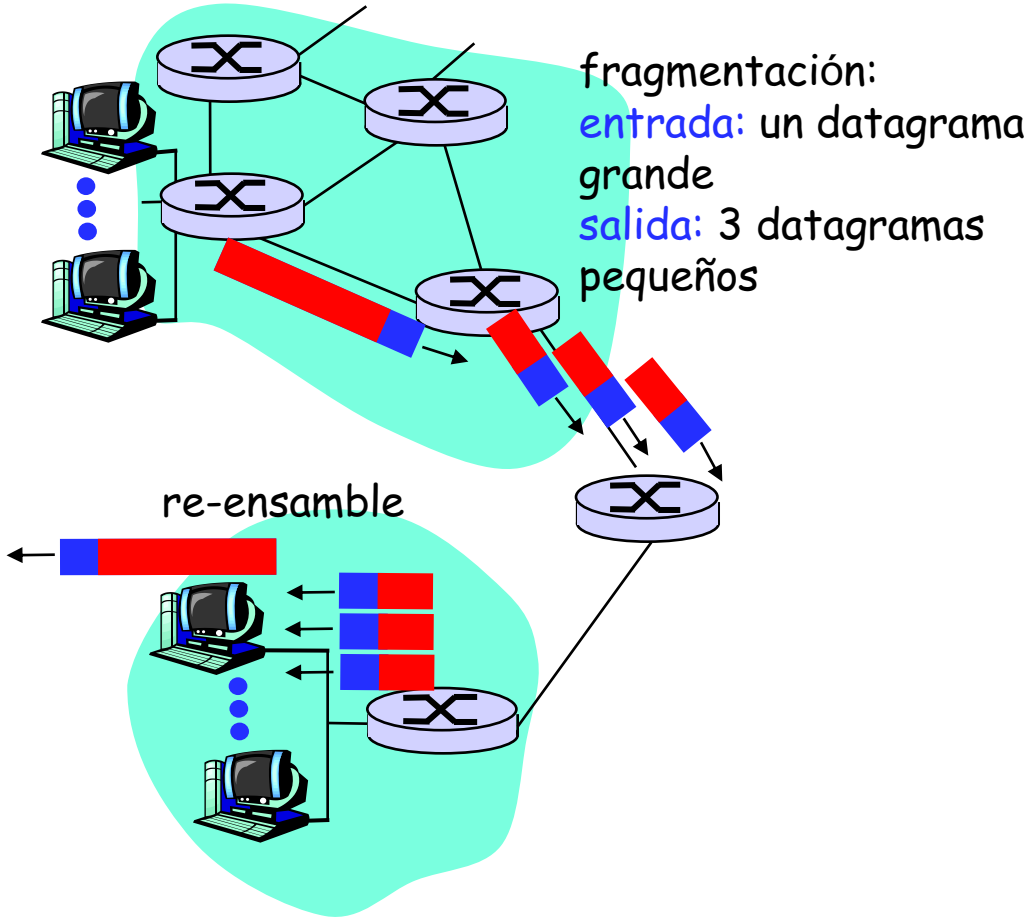


cuánto overhead con TCP?

- 20 bytes de TCP
- 20 bytes de IP
- = 40 bytes + overhead de la aplicación

Fragmentación & Re-ensamblado

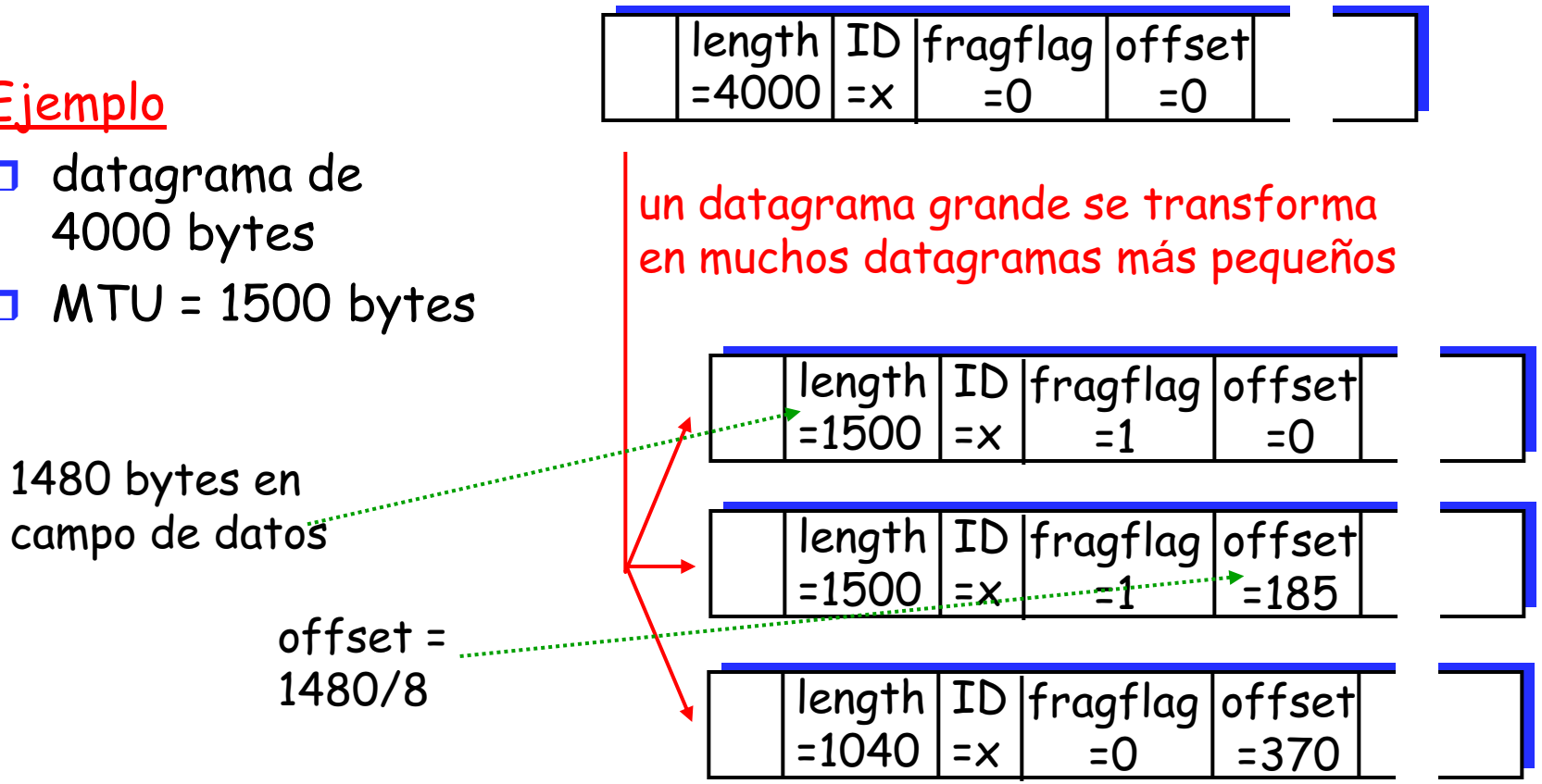
- los enlaces tienen un tamaño máximo de la unidad de trasmisión: MTU
  - enlaces diferents, MTUs diferentes
- los datagramas IP muy grandes pueden ser fragmentados en la red
  - un datagrama se transforma en muchos datagramas
  - "re-ensamble" en el destino
  - se usan bits del cabezal IP para identificar y ordenar los fragmentos



Fragmentación & Re-ensamblado

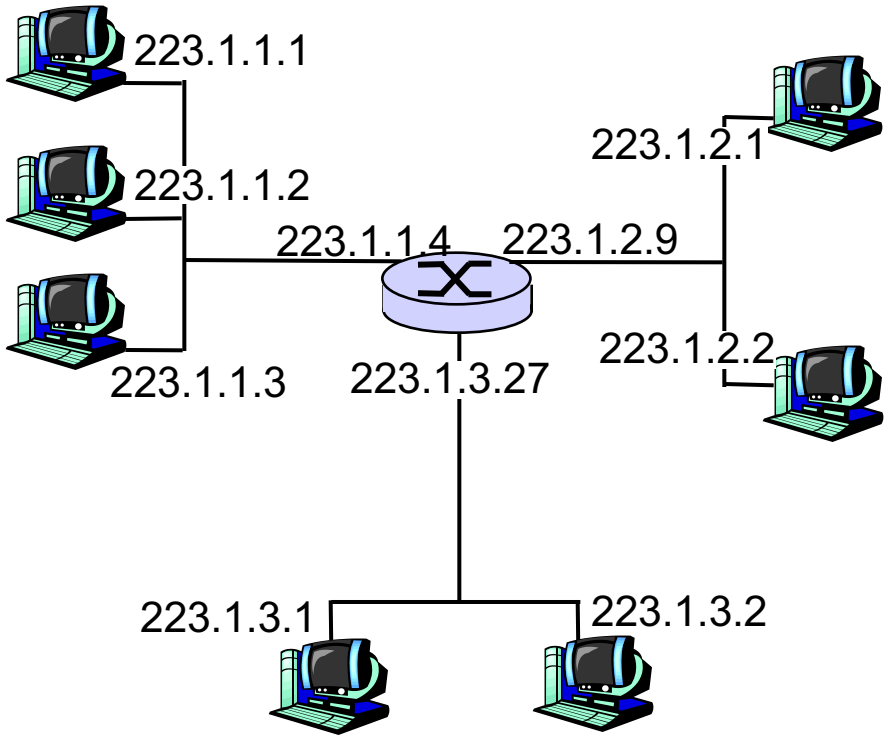
Ejemplo

- datagrama de 4000 bytes
- MTU = 1500 bytes



Direccionamiento IP: introducción

- dirección IP: identificador de 32-bit para una *interfaz* de host o router
- *interfaz:* conexión entre el host/router y el enlace físico
  - un router tiene típicamente muchas intefaces
  - un host tiene típicamente una sola interfaz
  - una dirección IP asociada a cada interfaz

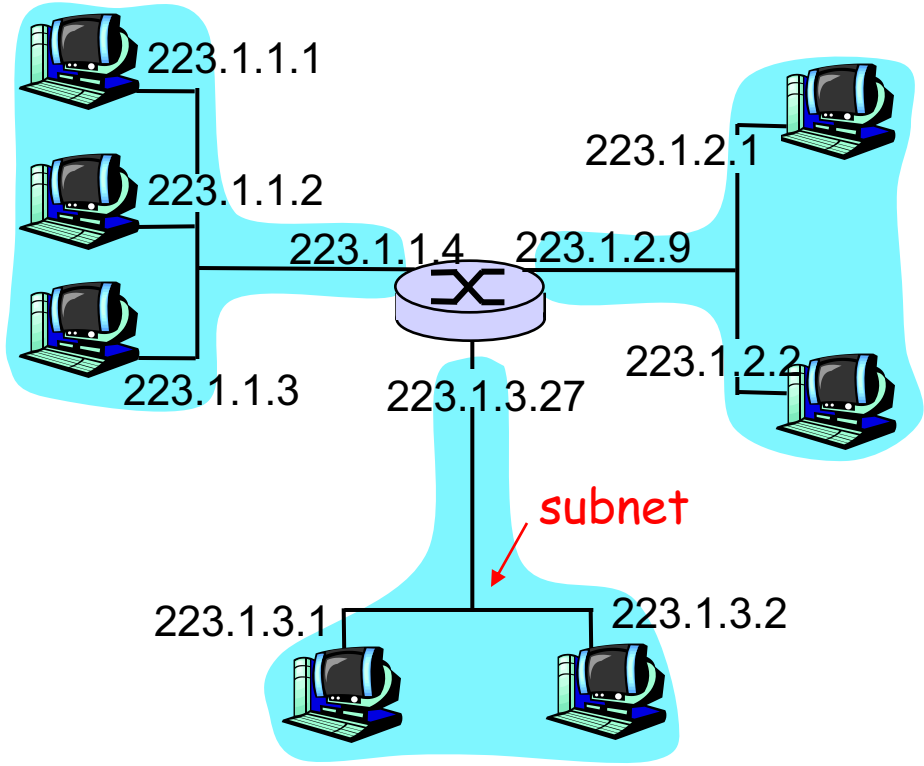


223.1.1.1 = 11011111 00000001 00000001 00000001

223 1 1 1

Subredes

- dirección IP:
  - subred (bits de mayor orden)
  - host (bits de menor orden)
- *qué es una subred?*
  - dispositivos cuya parte de subred de la dirección IP coincide...
  - ... pueden alcanzarce sin la intervención de un router (están en el "mismo cable", como una LAN hogareña)

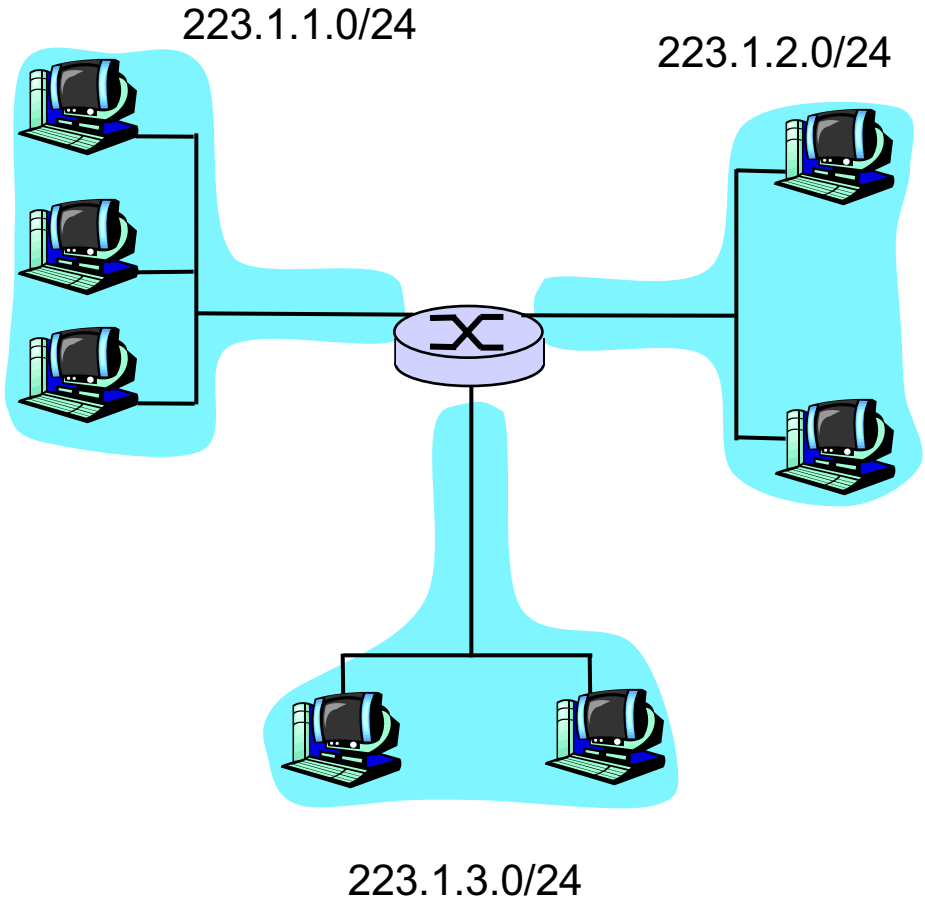


red integrada por 3 subredes

Subredes

"Receta"

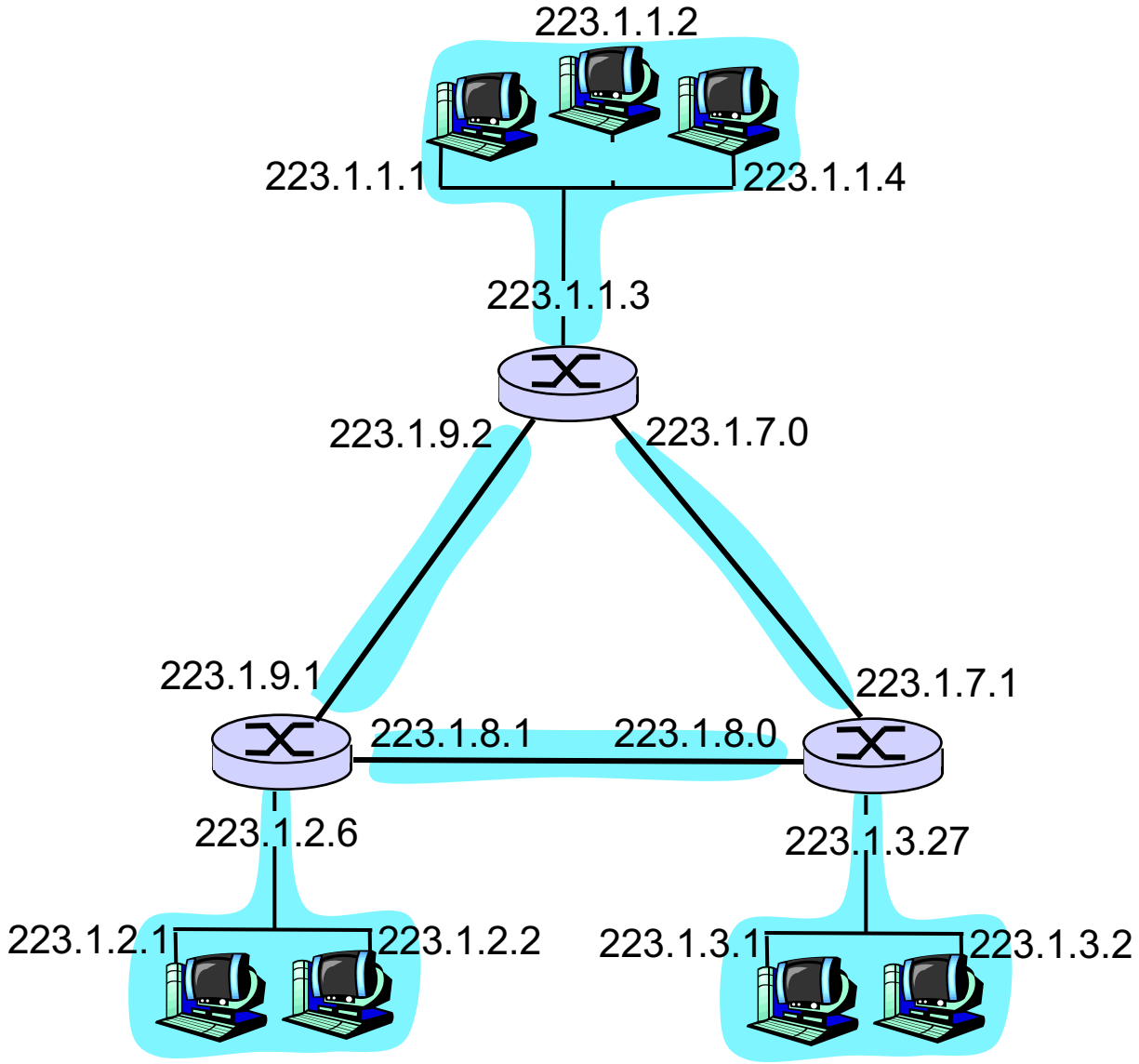
- Para determinar las subredes, desconectar cada interfaz de su host o router, creando islas de redes aisladas. Cada una de ellas en una subred.



Máscara de subred: /24

Subredes

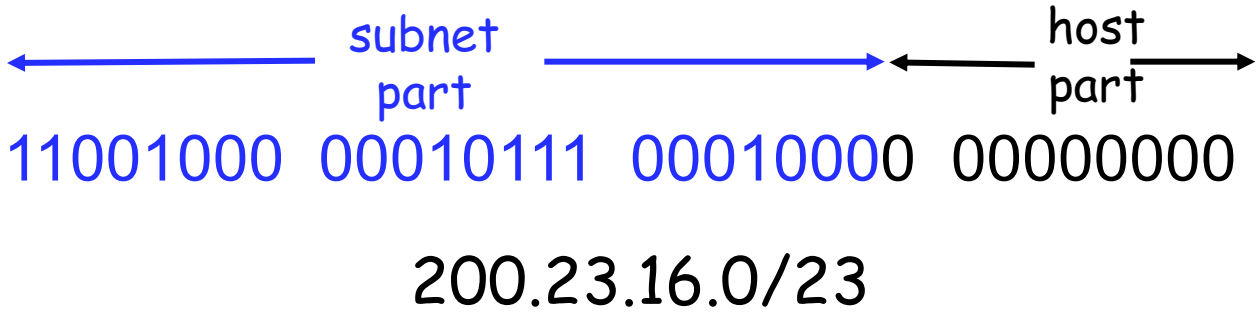
Cuántas?



direccionamiento IP: CIDR

CIDR: Classless InterDomain Routing

- porción de subred de la dirección de largo arbitrario
- formato de la dirección: a.b.c.d/x, donde x es el no. de bits en la parte de subred de la dirección



200.23.16.0/23

cómo obtener una dirección IP?

P: Cómo hace un host para obtener una dirección IP?

- "hard-coded" por el administrador de sistemas
  - Windows: "control-panel->network->configuration->tcp/ip->properties"
  - UNIX: /etc/rc.config o similar
- DHCP: Dynamic Host Configuration Protocol: obtención dinámica de una dirección, entregada por un servidor
  - "plug-and-play"

DHCP: Dynamic Host Configuration Protocol

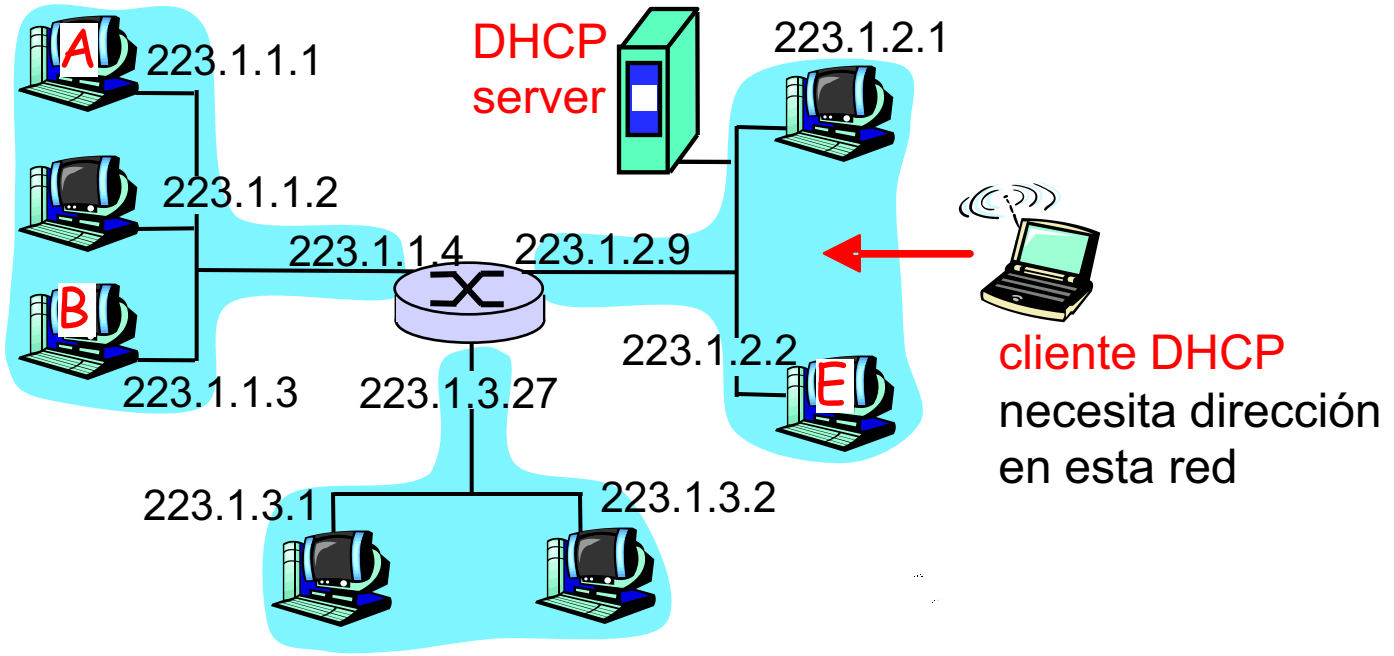
objetivo: permite a un host obtener una dirección IP *dinámicamente* de un servidor cuando se une a la red

- Renueva "lease" si la dirección está en uso
- Permite reuso de direcciones (solo se mantiene una dirección mientras el host está conectedo y "encendido")
- Soporte de usuarios móviles cuando "llegan" a una red

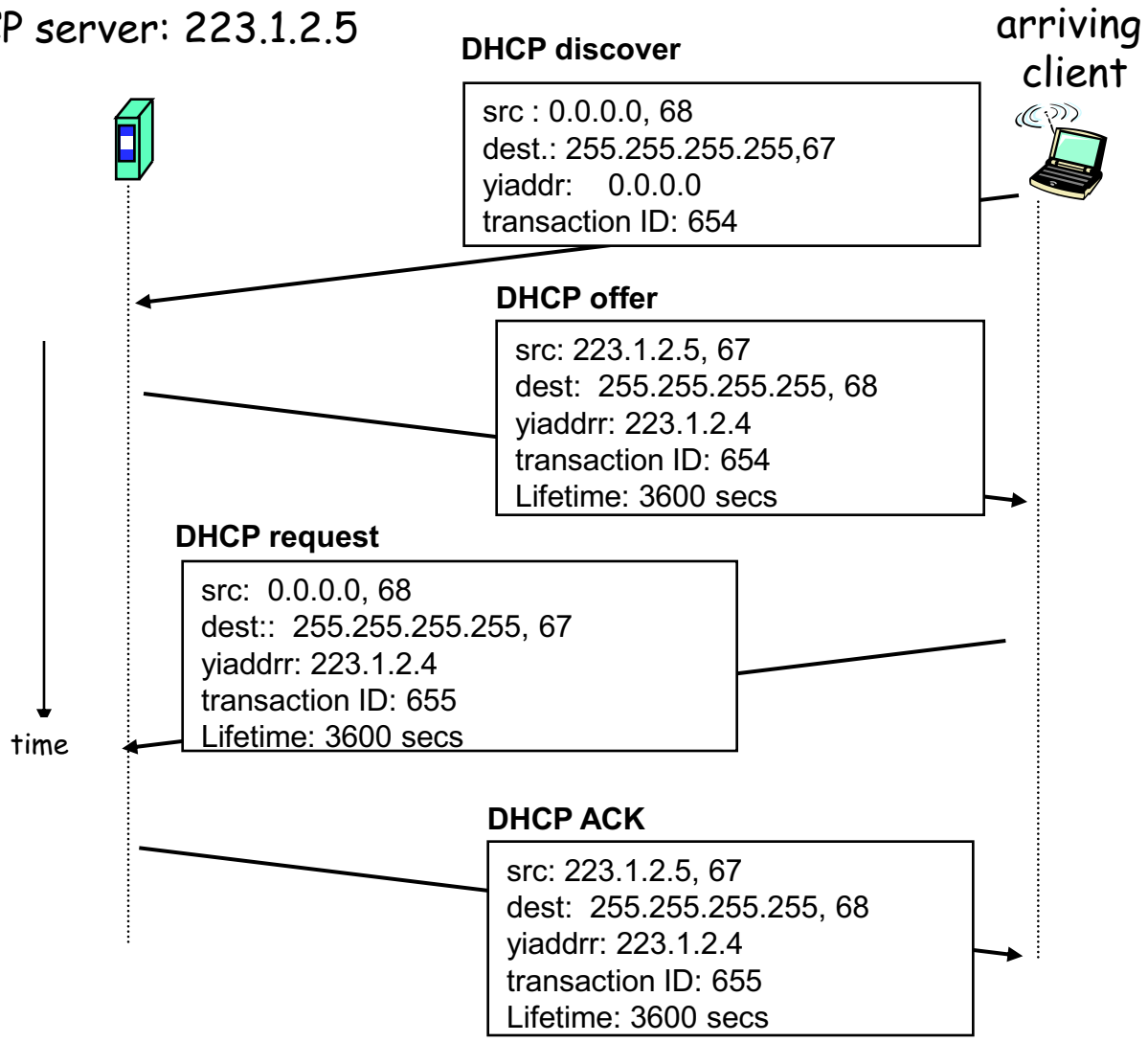
Carácterísticas del DHCP:

- host hace broadcast de mensaje "DHCP discover"
- servidor DHCP responde con mensaje "DHCP offer"
- host pide una dirección IP con el mensaje "DHCP request"
- servidor DHCP envía dirección en mensaje "DHCP ack"

Escenario DHCP cliente-servidor

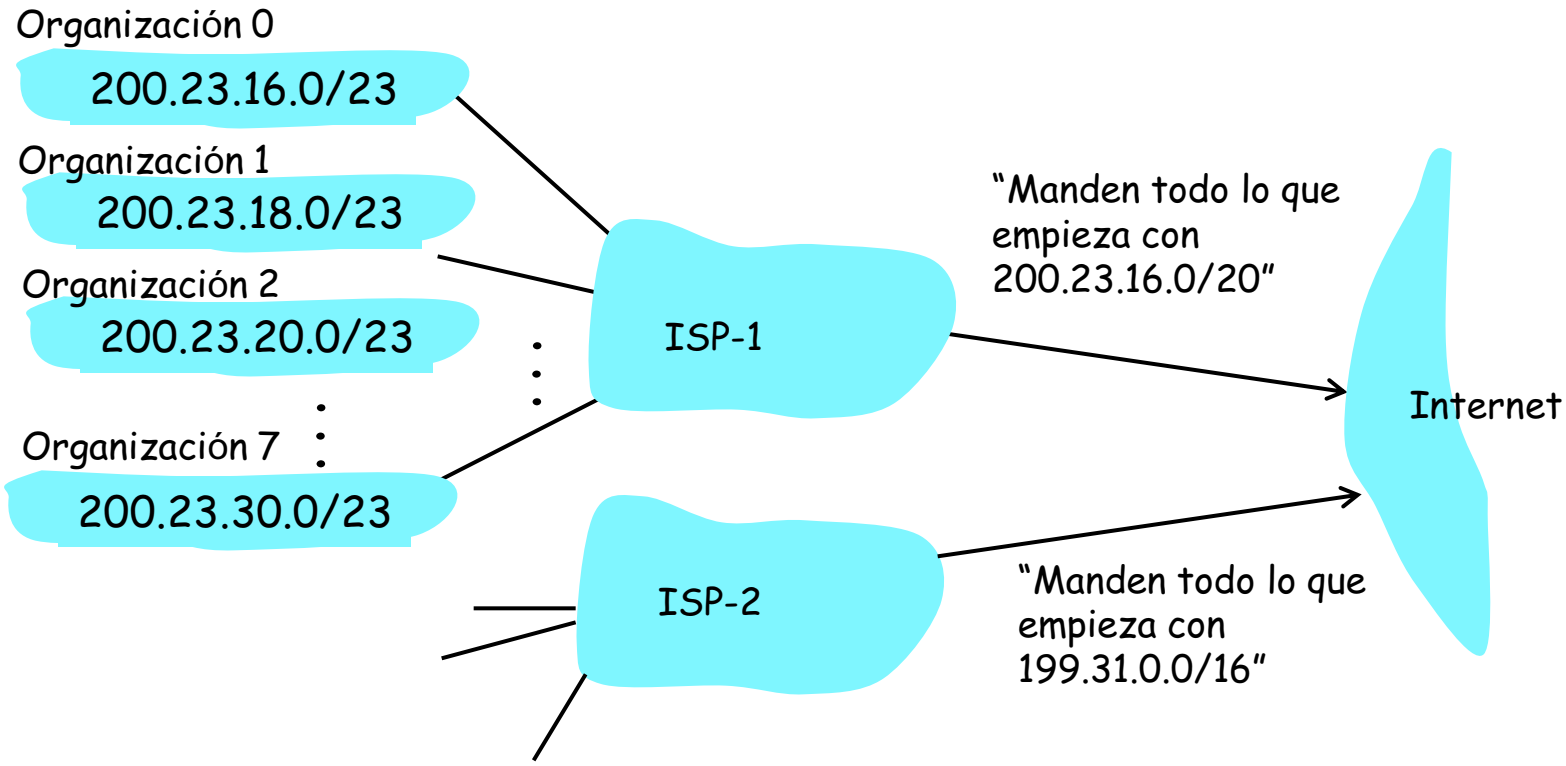


Escenario DHCP cliente-servidor



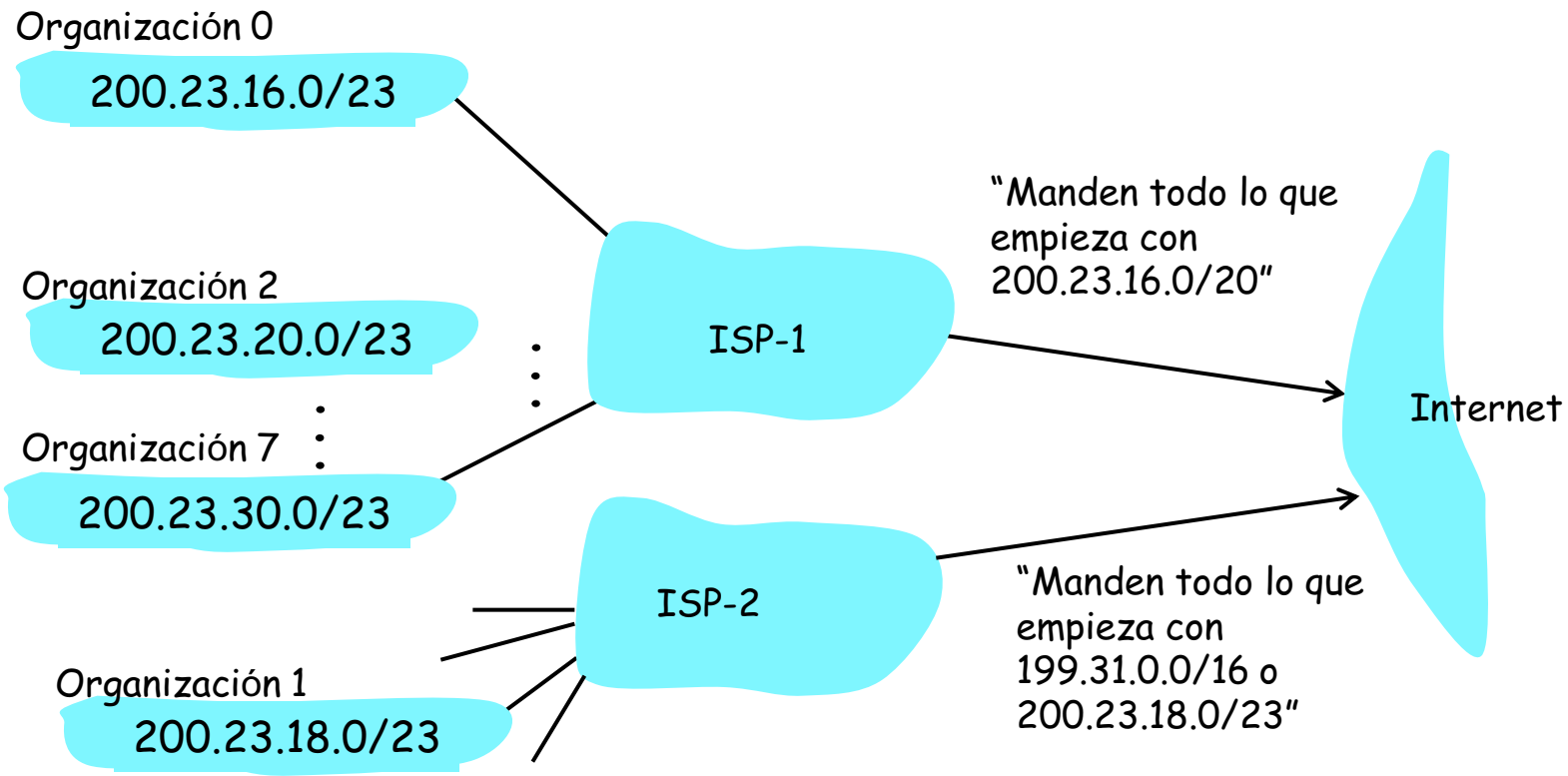
Direccionamiento jerárquico: agregación de rutas

El direccionamiento jerárquico permite publicar en forma eficiente la información de enrutamiento:



Direccionamiento jerárquico: rutas más específicas

Organización 1 se mueve de ISP-1 a ISP-2



"gobierno" de la internet…

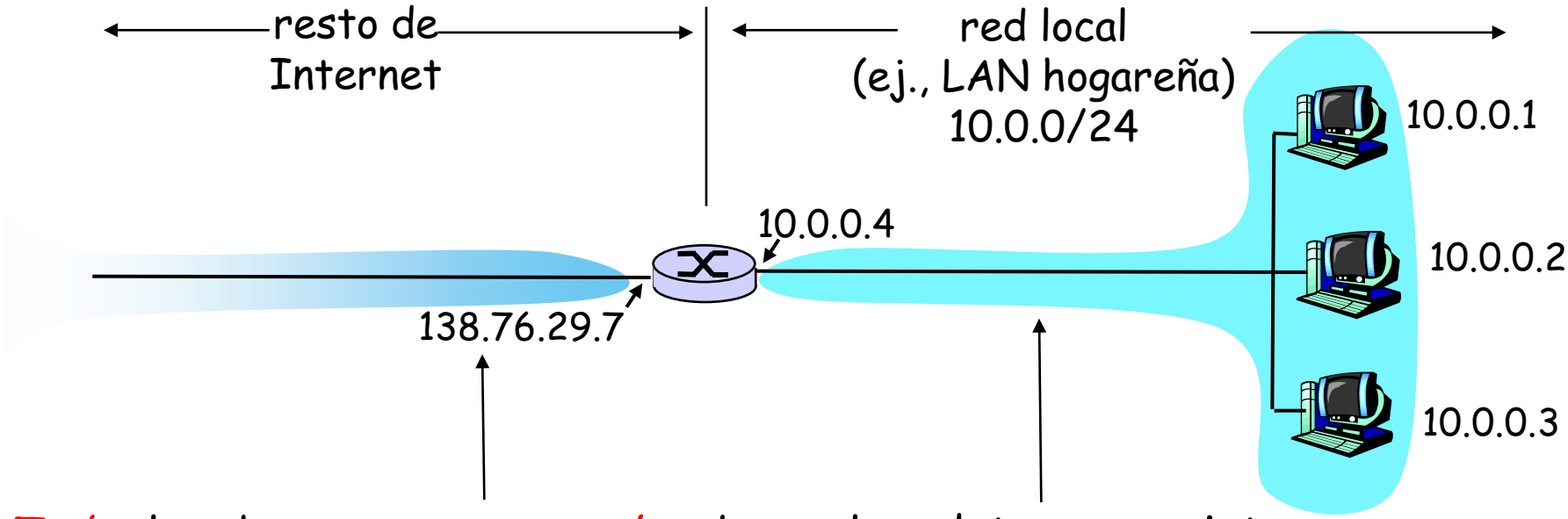
P: Cómo obtiene un ISP un bloque de direcciones?

R: ICANN: Internet Corporation for Assigned Names and Numbers

- asignar direcciones
- gestionar DNS
- asignar nombres de dominio, resolver disputas

- Regionalización
  - AFRINIC, RIPE NCC, ARIN, APNIC, LACNIC

NAT: Network Address Translation



Todos los datagramas *que salen* de la red local tienen la misma dirección IP: 138.76.29.7, se diferencian los puertos de origen

Los datagramas internos a la red tiene la dirección 10.0.0/24 de fuente/destino (como siempre)

NAT: Network Address Translation

- Motivación: la red local utiliza una sola dirección IP visto desde el mundo exterior:
  - no es necesario solicitar un rango de direcciones al ISP: solo una dir. IP para todos los dispositivos
  - se pueden cambiar direcciones de los dispositivos en la red local sin notificar al "resto del mundo"
  - Se puede cambiar de ISP sin modificaciones en la red local
  - los dispositivos en la red local no son "visibles" desde el mundo exterios (un extra de seguridad).

NAT: Network Address Translation

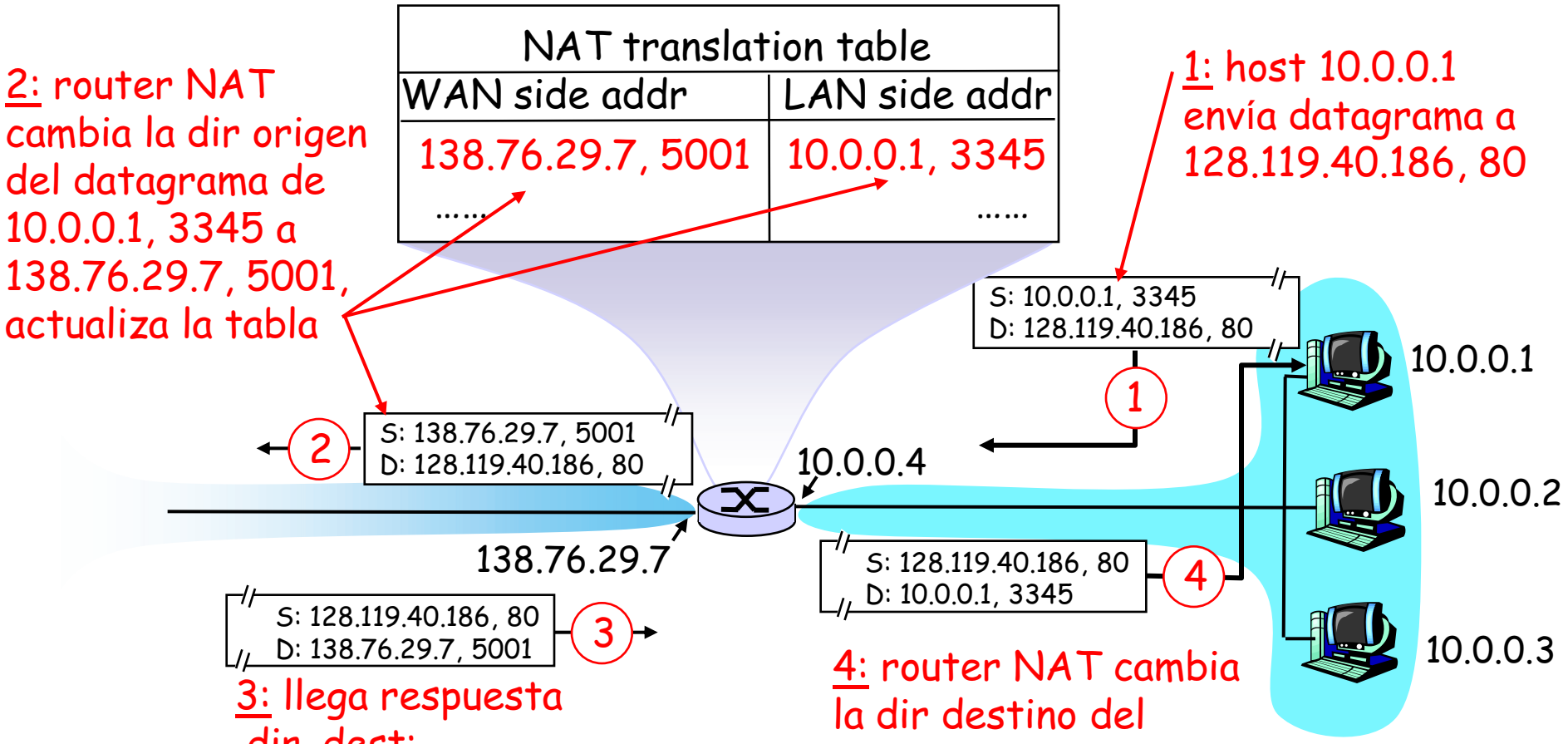
Implementación: un router NAT debe:

- *datagramas salientes: reemplazar* (dir IP origen, port #) de cada datagrama a (dir IP del NAT, new port #)
  . . . clientes/servidores remotos responderán usando (NAT IP address, new port #) como destino.

- *recordar (en la NAT translation table)* cada par de traslaciones (dir IP origen, port #), (dir IP del NAT, new port #)

- *datagramas entrantes: reemplazar* (dir IP NAT, new port #) en campos destino al correspondiente (dir IP origen, port #) almancenado en la tabla de NAT

NAT: Network Address Translation

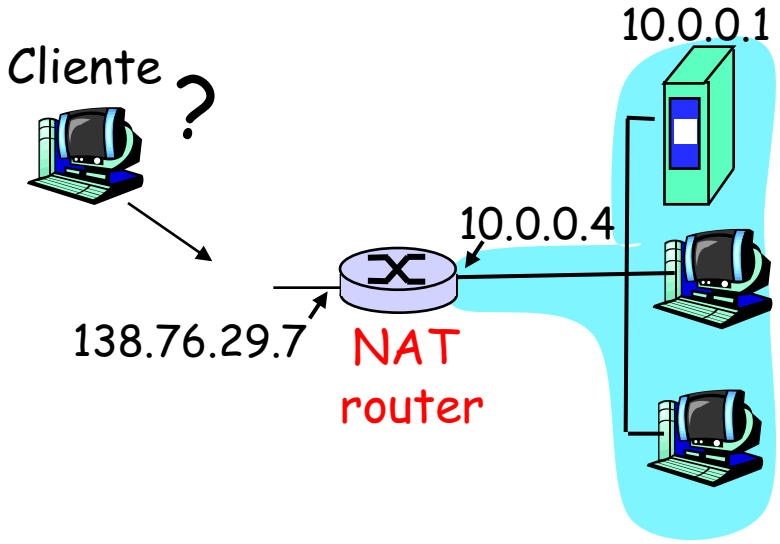


NAT: Network Address Translation

- campo *port-number* de 16 bits:
  - 60,000 conexiones simultáneas con una sola dirección IP!
- NAT es contradictorio:
  - los routers solo deberían procesar hasta capa 3
  - viola la abstracción de extremoa extremo
    - el NAT debe ser tenido en cuenta por los diseñadores de aplicaciones, por ej. P2P
  - la carencia de direcciones debería resolverse por métodos más “limpios”, como IPv6

como atravesar un NAT?

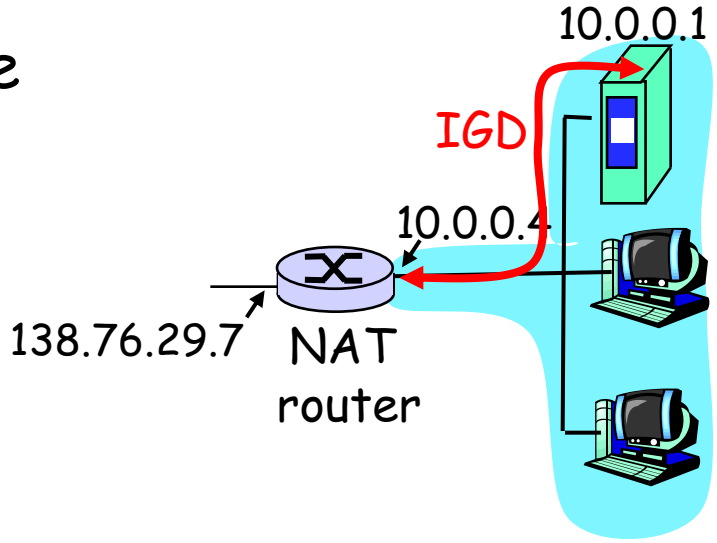
- un cliente se quiere conectar al servidor con dirección 10.0.0.1
  - pero esa dir. es local a la LAN, no se puede usar como dir. de destino
  - Solo se puede usar la dir del NAT: 138.76.29.7
- solución 1: configuración estática de *port forwarding*
  - ej., (123.76.29.7, port 2500) siempre se traduce a 10.0.0.1 port 25000



como atravesar un NAT?

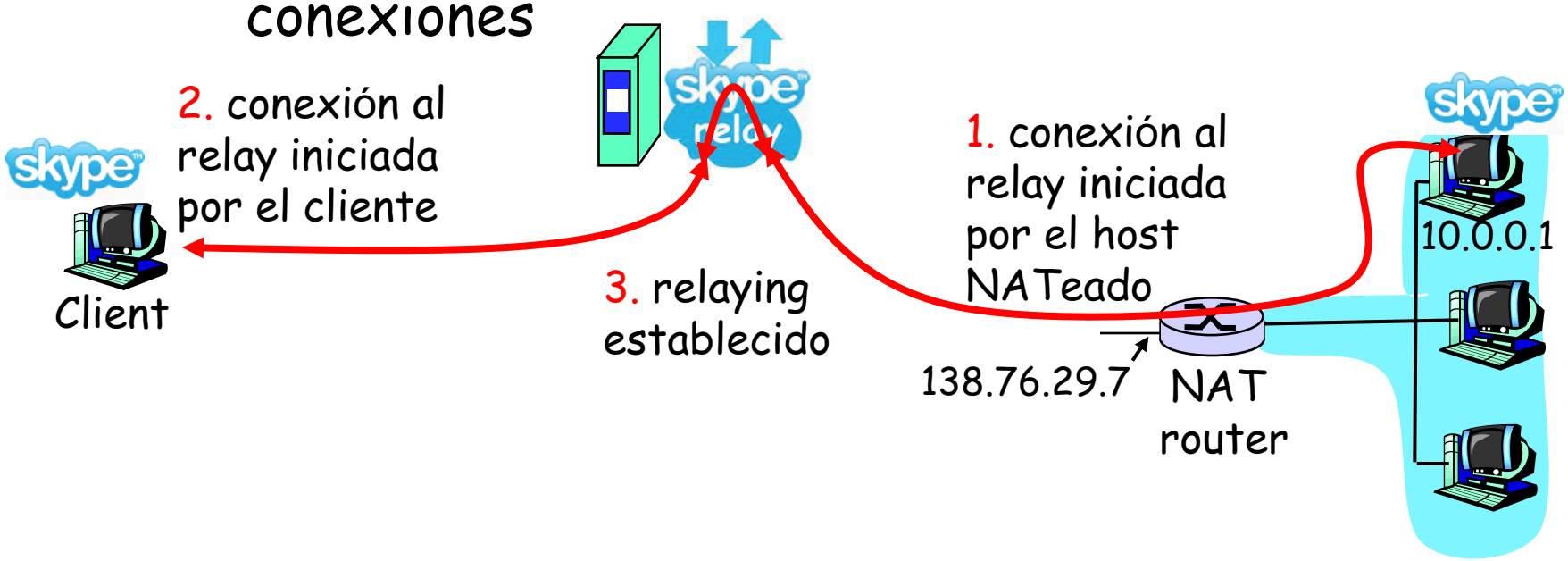
- solución 2: Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Permite a un host detrás de un NAT a:
  - aprender la dirección IP pública (138.76.29.7)
  - agregar/remover mapeos de puertos (con tiempos de lease)

  es decir, automatiza la configuración estática del port forwarding del NAT



como atravesar un NAT?

- solución 3: "relaying" (usado en Skype)
  - cliente "NATeado" establece conexión al relay
  - clientes externos se conectan al relay
  - relay hace "bridge" de paquetes entre conexiones



ICMP: Internet Control Message Protocol

- usado por hosts & routers para comunicar informaciónd e nivel de red
  - reporte de errores: *unreachable* host, red, puerto, protocolo
  - echo request/reply (usado por el ping)
- capa de red "encima" de IP:
  - mensajes ICMP transportado en datagramas IP
- mensajes ICMP: tipo, código, más los primeros 8 bytes del datagrama IP qué causó el error

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute e ICMP

- fuente envía una serie de segmentos UDP al destino
  - el 1o tiene TTL =1
  - el 2o tiene TTL=2, etc.
  - no. de puerto "raro"
- cuando el n-simo datagrama arriba al n-simo router:
  - el router descarta el datagrama…
  - …y envía a la fuente un mensaje ICMP (type 11, code 0)
  - el mensaje incluye el nombre y dir IP del router
- cuando llega el mensaje ICMP, la fuente calcula el RTT
- traceroute repite esta operación 3 veces

criterio de parada

- el segmento UDP eventualmente llega al host destino
- este retorna el mensaje ICMP "host unreachable" (type 3, code 3)
- cuando la fuente recibe este paquete ICMP, para.

Cap. 4: Capa de red

- 4. 1 Introducción
- 4.2 circuitos virtuales y datagramas
- 4.3 dentro de un router
- 4.4 IP: Internet Protocol
  - formato de datagramas
  - direccionamiento IPv4
  - ICMP
  - IPv6
- 4.5 Algoritmos de enrutamiento
  - Link state
  - Distance Vector
  - Enrutamiento jerárquico
- 4.6 Enrutamiento en Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast y multicast

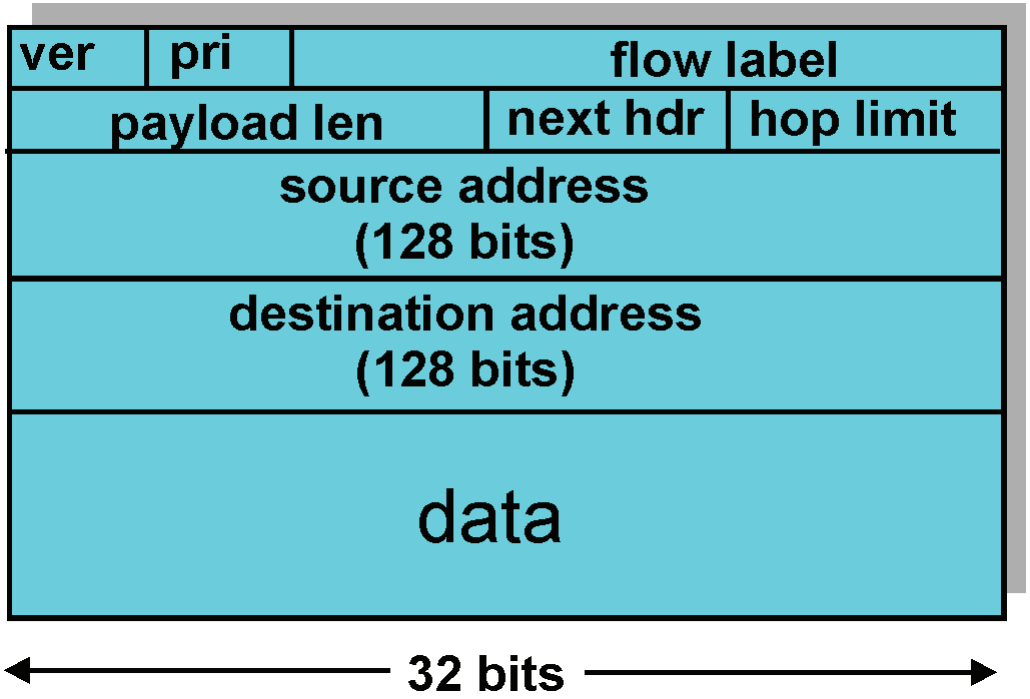
IPv6

- motivación inicial: el espacio de direcciones de 32 bits "se está por agotar.
- motivación adicional:
  - formato del cabezal ayuda a acelerar el procesamiento/forwarding del paquete
  - cambios en el cabezal facilitan QoS

  formato del datagrama IPv6:
  - cabezal de largo fijo: 40 bytes
  - no se permite fragmentación

Cabezal IPv6

Priority: identifica prioridad entre datagramas en un flujo
Flow Label: identifica datagramas en el mismo flujo (concepto de "flujo"...).
Next header: identifica el protocolo de capa superior



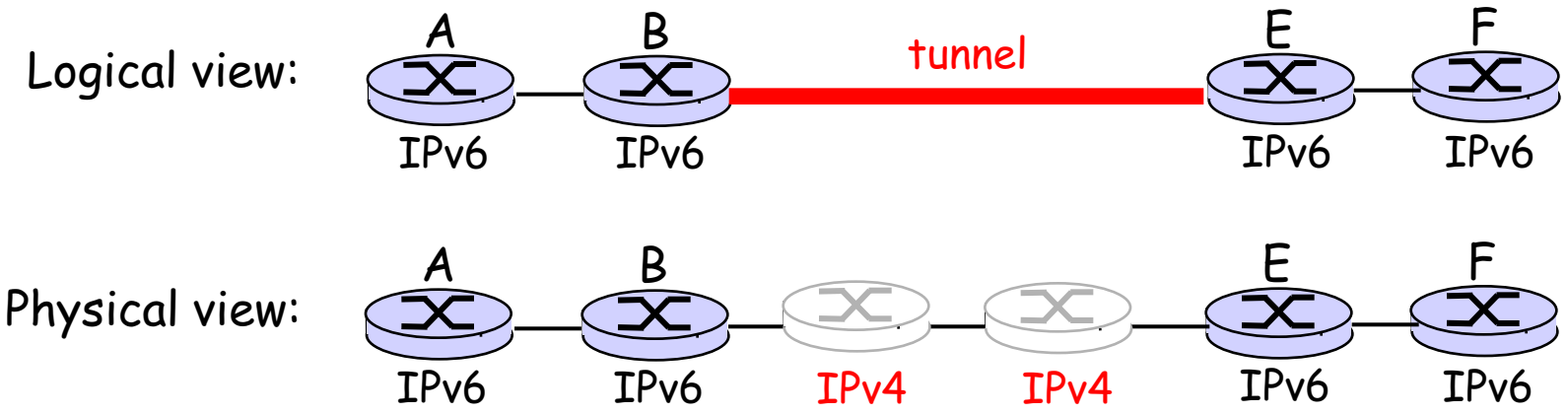
Otros cambios con respecto a IPv4

- *Checksum*: eliminado para reducir procesamiento en cada hop
- *Options:* permitido, pero fuera del header, indicado por el campo "Next Header"
- *ICMPv6:* nueva versión de ICMP
  - Tipos de mensajes adicionales, por ej. "Packet Too Big"
  - Funciones de gestión de grupos de multicast

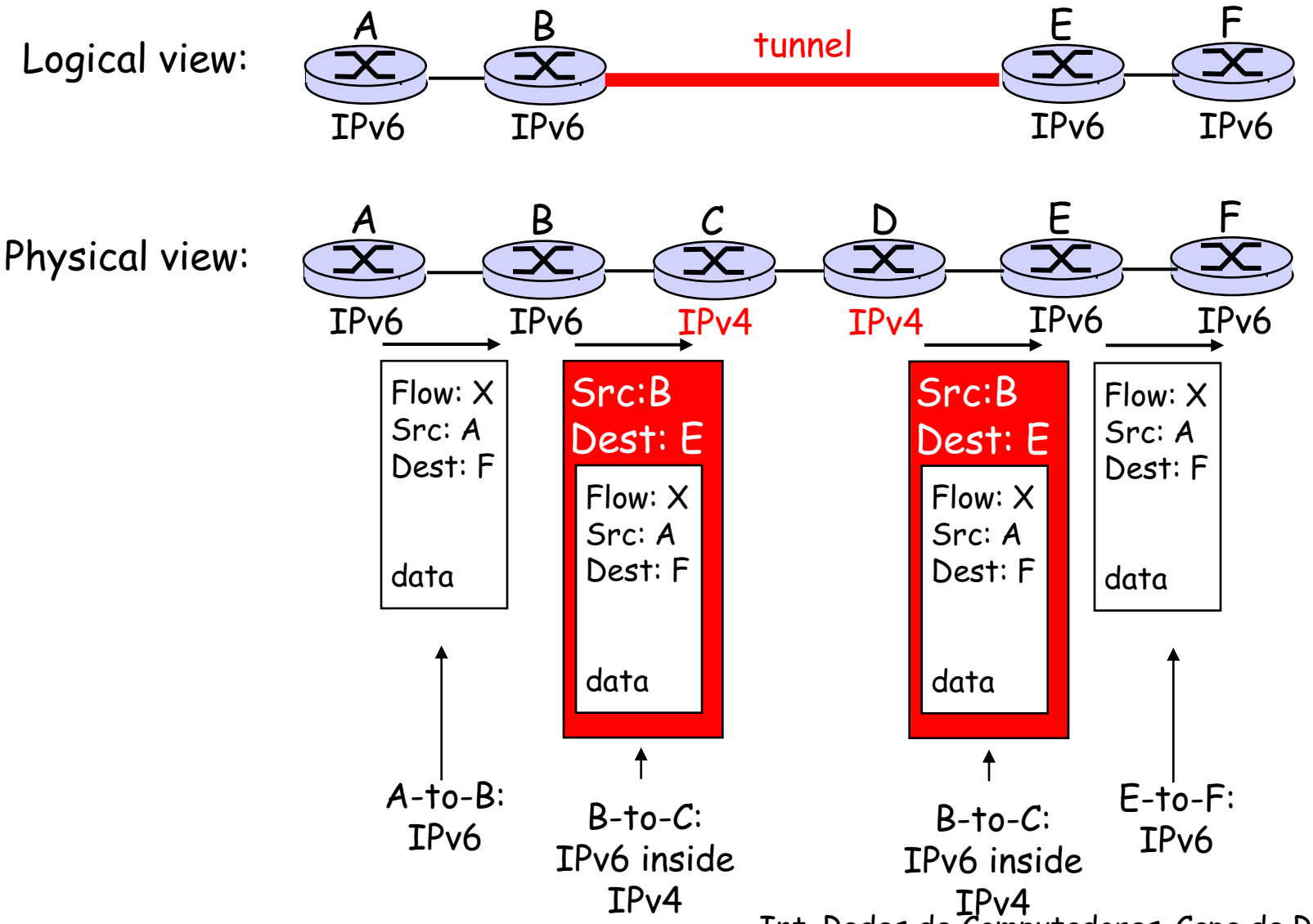
Transición de IPv4 a IPv6

- No se puede hacer una actualización de todos los routers simultáneamente
  - no hay "dia D"
  - Cómo puede operar una red con routers IPv4 e IPv6 mezclados?
- *Tunneling:* IPv6 transportado como *payload* en datagramas IPv4

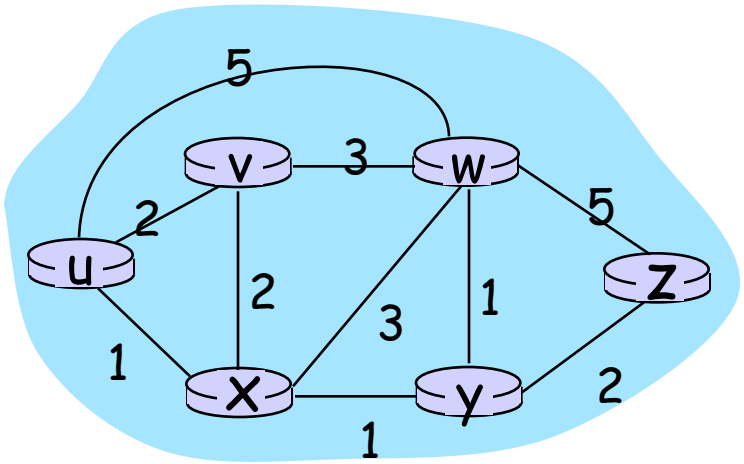
Tunneling


Logical view:
A
B
tunnel
E
F
IPv6
IPv6
IPv6
IPv6
Physical view:
A
B
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6

Tunneling


Logical view:
A
B
tunnel
E
F
IPv6
IPv6
IPv6
IPv6
Physical view:
A
B
C
D
E
F
IPv6
IPv6
IPv4
IPv4
IPv6
IPv6
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Src:B
Dest: E
Flow: X
Src: A
Dest: F
data
Flow: X
Src: A
Dest: F
data
A-to-B:
IPv6
B-to-C:
IPv6 inside
IPv4
B-to-C:
IPv6 inside
IPv4
E-to-F:
IPv6

Abstracción de grafo



Graph: G = (N,E)

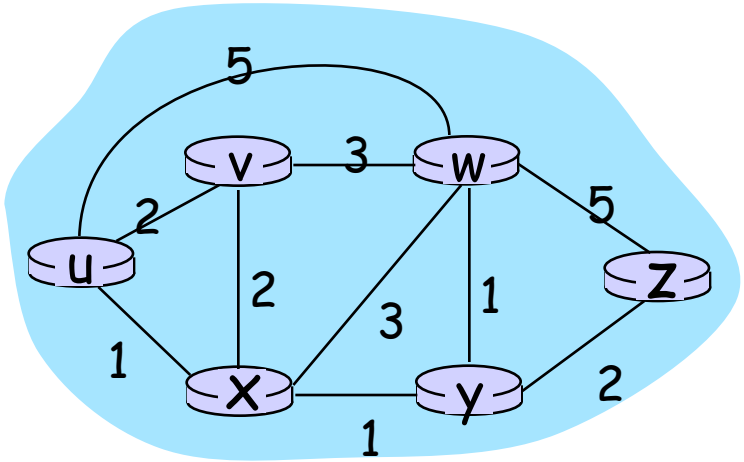
N = conjunto de routers = { u, v, w, x, y, z }

E = conjunto de enlaces ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Nota: esta abstracción es útil en otros contextos de red

Ejemplo: P2P, donde N es el conjunto de pares y E el conjunto de conexiones TCP

Abstracción de grafo: costos



- c(x,x') = costo del enlace (x,x')

  - por ej., c(w,z) = 5

- algunas opciones para el costo: puede ser 1, con relación inversa al ancho de banda, con relación inversa a la congestión, entre otras

costo del camino $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

pregunta: cual es el camino con menor costo entre u y z ?

Algoritmo de enrutamiento:
encuentra el camino de costo mínimo

Clasificación de algoritmos de enrutamiento

Información global o descentralizada?

Global:

- todos los routers conocen la topología completa, y el costo de los enlaces
- algoritmos "link state"

Descentralizada:

- los router conocen los vecinos directamente conectados, y el costo de los enlaces a estos vecinos
- proceso de cómputo iterativo, con intercambio de información entre vecinos
- algoritmos "distance vector"

Estático o dinámico?

Estático:

- las rutas cambian lentamente

Dinámico:

- Cambios más frecuentes en rutas
  - actualización periódica
  - en respuesta a cambios en topología o costo de los enlaces

Algoritmo de enrutamiento Link-State

Algoritmo de

- Topología de la red y costos de los enlaces conocidos por todos los nodos
  - mediante "link state advertisements"
  - todos los nodos Dijkstra tienen la misma información
- se computan los caminos de costo mínimo entre un nodo (raíz) al resto de los nodos
  - determina la tabla de forwarding para ese nodo
- iterativo: luego de k iteraciones, se conocen los caminos de costo mínimo a k destinos

Notación:

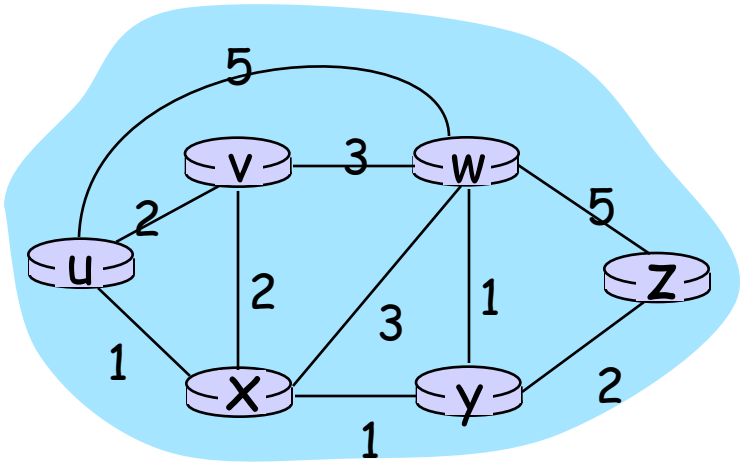
- c(x,y): costo del enlace entre nodos x,y; = ∞ si no son vecinos directos
- D(v): valor actual del costo del camino desde origen al destino v
- p(v): nodo predecesor en el camino desde fuente a destino v
- N': conjunto de nodos cuyo costo de camino se ha computado

Algoritmo de Dijsktra

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

Algoritmo de Dijkstra: ejemplo

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Algoritmo de Dijkstra: ejemplo (cont)

"Shortest-path tree" resultante desde u:

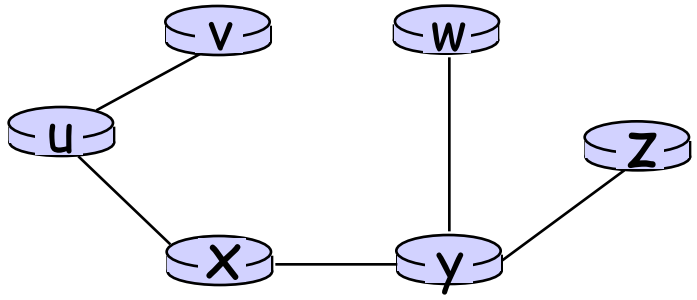


Tabla de forwarding resultante en u:

destino	enlace
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

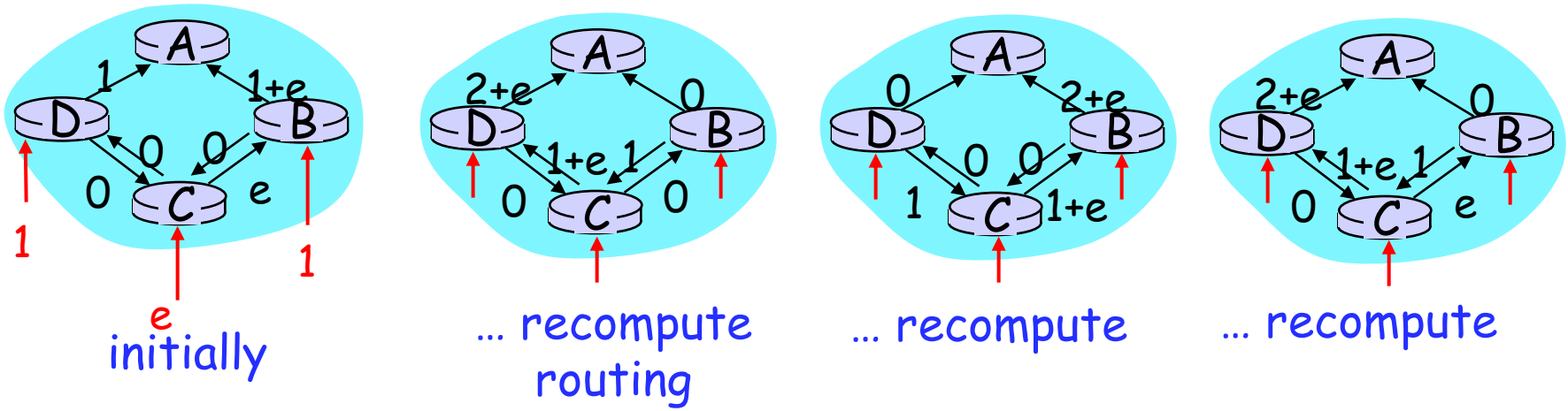
Algoritmo de Dijkstra: discusión

Complejidad del algoritmo: n nodos

- cada iteración: necesita chequear todos los nodos, w, que no están en N
- n(n+1)/2 comparaciones: $O(n^2)$
- implementación más eficiente posible: O(nlogn)

Posibles oscilaciones:

- por ej., costo del enlace = cantidad de tráfico transportado



Algoritmo Distance Vector

Ecuación de Bellman-Ford (programación dinámica)

Se define

$d_x(y)$:= costo del camino de menor costo de x a y

Luego

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

donde min se calcula entre todos los vecinos v de x

Algoritmo Distance Vector

- $D_x(y)$ = estimación del menor costo de x a y
- Nodo x conoce el costo a cada vecino v: $c(x,v)$
- Nodo x mantiene el "distance vector" $\mathbf{D_x}$ = $[D_x(y): y \in N]$
- Nodo x también mantiene el vector de distancia hacia sus vecinos
  - Para cada vecino v, x mantiene $\mathbf{D_v} = [D_v(y): y \in N]$

Algoritmo Distance Vector

Idea básica:

- Cada cierto tiempo, cada nodo envía su estimación de "distance vector" a sus vecinos
- Asincrónico
- Cuando un nodo x recibe una nueva estimación del DV de su vecino, actualiza su propio DV usando la ecuación de B-F:

$D_x(y) \leftarrow min_v\{c(x,v) + D_v(y)\}$ *para cada nodo* $y \in N$

- Bajo condiciones "naturales", la estimaación $D_x(y)$ *converge al menor costo* $d_x(y)$

Algoritmo Distance Vector

Iterativo, asincrónico:

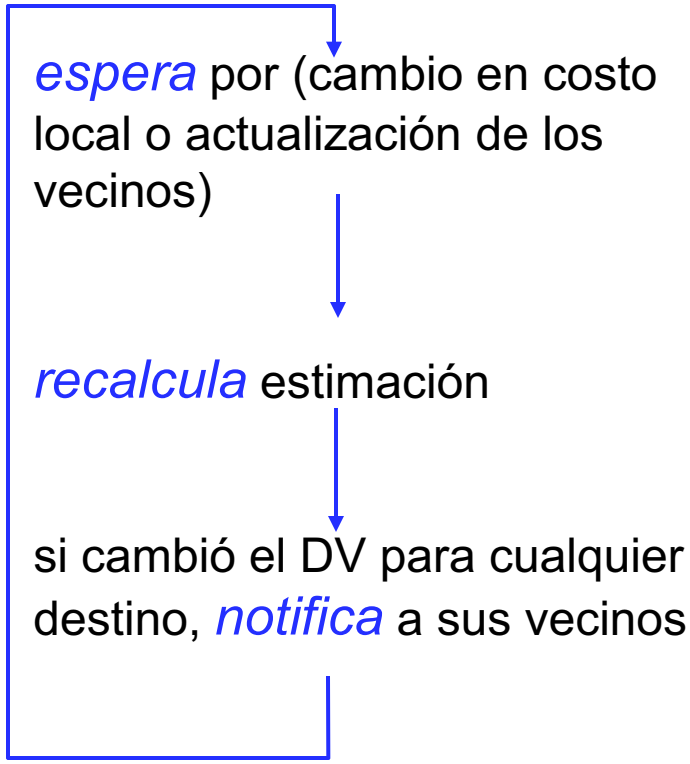
cada iteración local causada por:

- cambio en el costo de enlaces local
- mensaje de actualización del DV de un vecino

Distribuido:

- cada nodo notifica a sus vecinos *solo* cuando cambia su DV
  - vecinos notifican luego a sus vecinos si es necesario

Cada nodo:



Algoritmo Distance Vector

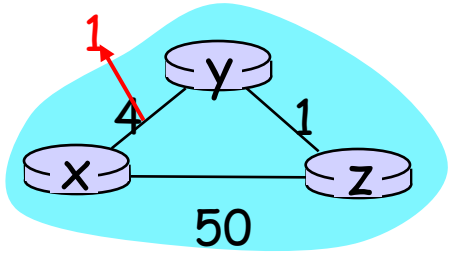
en todos los nodos, X:

```
1  Initialization:
2   for all destinations y in N:
3      D_X(y) = c(X,y) /* if y is not a neighbor c(X,y) = ∞ */
4   for each neighbor w
5      D_X(y) = ∞ for all destinations y in N
5   for each neighbor w
6      send distance vector D_X = [D_X(y): y in N] to w
7
9  loop
10    wait (until I see a link cost change to some neighbor w
11          or until I receive DV update from neighbor w)
12
13   for each y in N:
14     D_X(y) = min {c(X,v) + D_X(y)}
15
16   if D_X(y) changed for any destination y
17     send distance vector D_X = [D_X(y): y in N] to w
18
19  forever
```

Distance Vector: cambios en los costos

Cambia el costo de un enlace:

- nodo detecta el cambio
- actualiza información de routing, recalcula distance vector
- si cambia DV, notifica a vecinos



"las buenas noticias viajan rápido"

En tiempo t_0, *y* detecta el cambio de costo, actualiza su DV, e informa a sus vecinos.

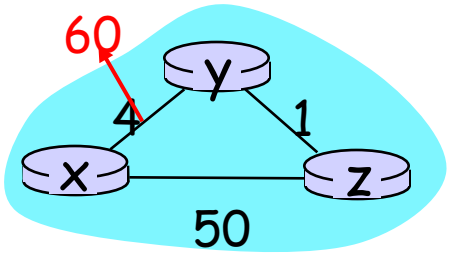
En tiempo t_1, *z* recibe la actualización desde *y*, actualiza su tabla. Calcula el nuevo cost mínimo a *x*, envía su DV a los vecinos.

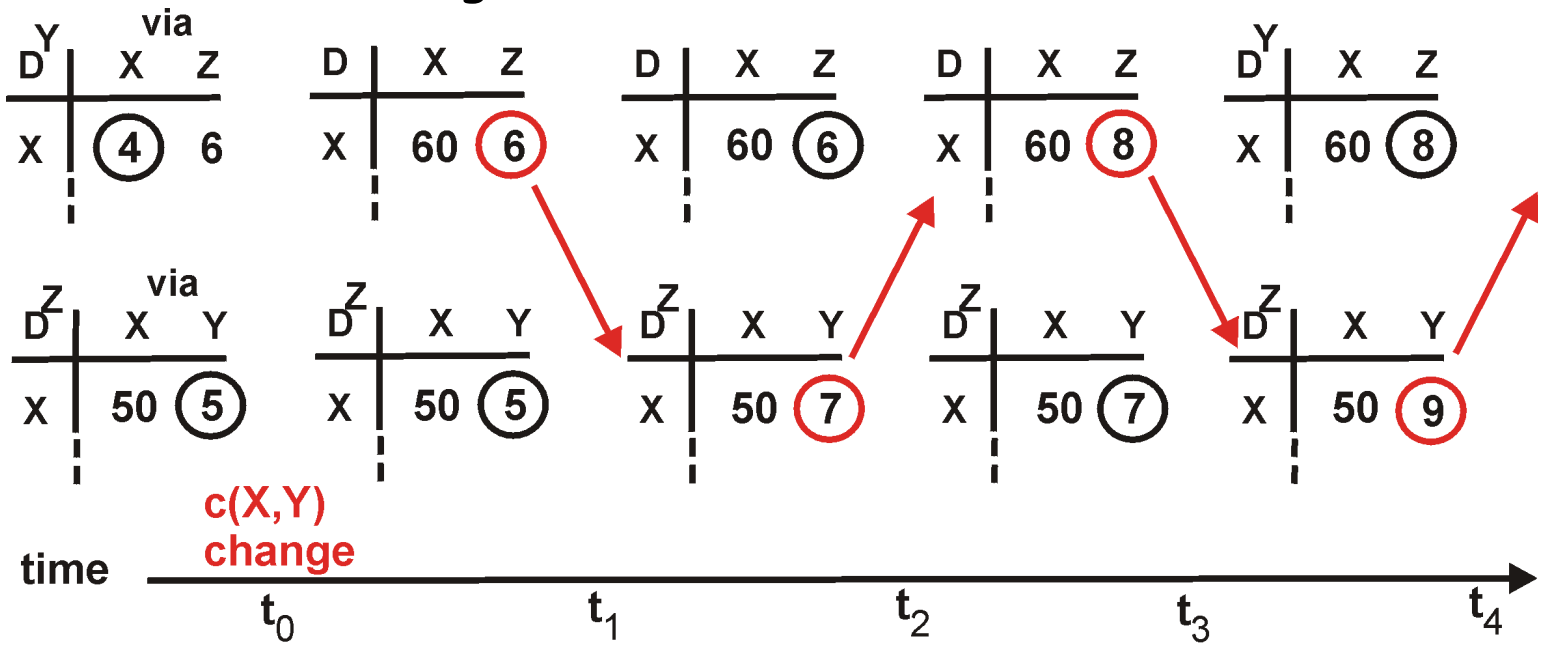
En tiempo t_2, *y* recibe la actualización desde *z*, actualiza su tabla de distancias.
El costo mínimo de *y* no cambia, entonces *no* envía ninguna actualización

Distance Vector: cambios en los costos

Cambia el costo de un enlace:

- buenas noticias viajan rápido
- malas noticias viajan lento - problema "count to infinity"!
- 44 iteraciones para que se estabilice el algoritmo





Enrutamiento Jerárquico

Hasta ahora hemos visto una idealización

- routers idénticos
- red "plana"

… *no* sucede en la práctica

escala: 100+ millones de destinos:

- no es posible almacenar todos los destinos en las tablas de routing!
- Los intercambios de información inundarían los enlaces!

autonomía administrativa

- internet = red de redes
- cada administrador quiere tener control de su propia red

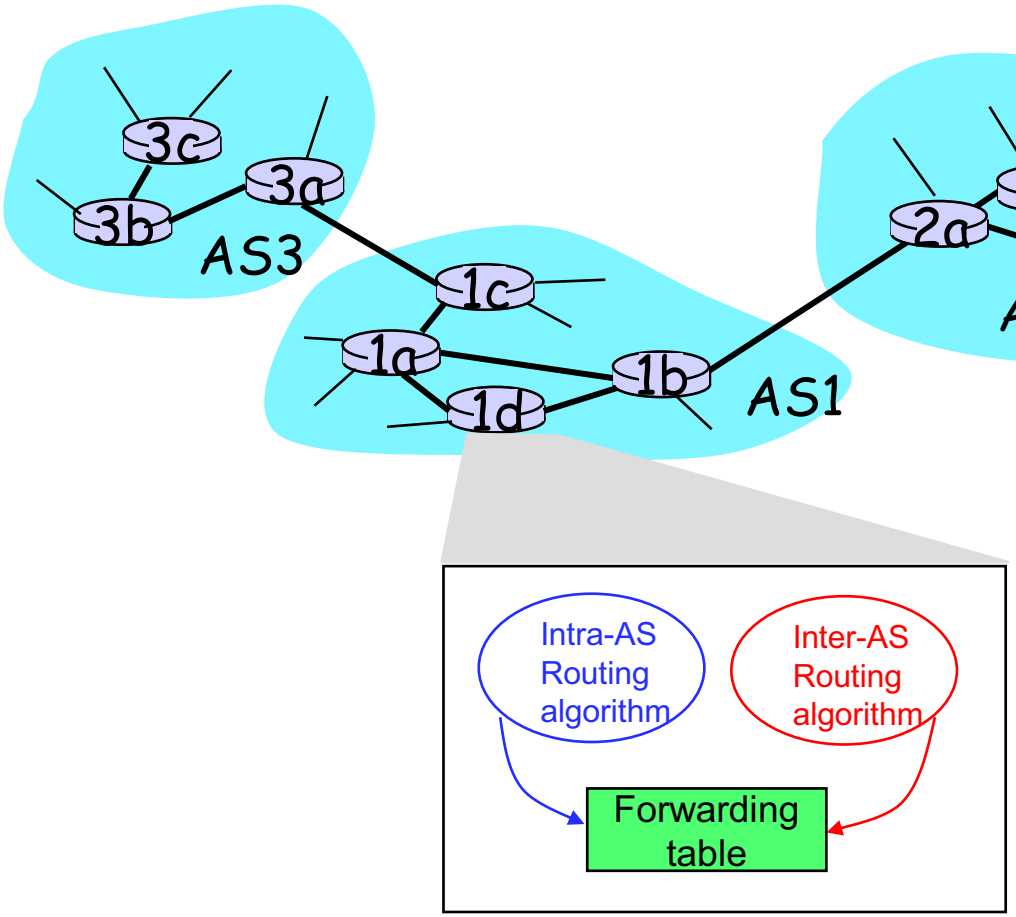
Enrutamiento Jerárquico

- agrupamiento de routers en regiones, "autonomous systems" (AS)
- routers dentro de un AS corren el mismo protocolo de routing
  - "intra-AS" routing protocol
  - routers en ASs diferentes pueden usar protocolos de routing diferentes

"gateway" router

- Enlace(s) directo(s) a router(s) en otro(s) AS(s)

Interconexión de ASs



- la tabla de forwarding se configura usando los protocolos de routing intra- e inter-AS
  - intra-AS configuran entradas para destinos internos
  - inter-AS & intra-As configuran entradas para destinos externos

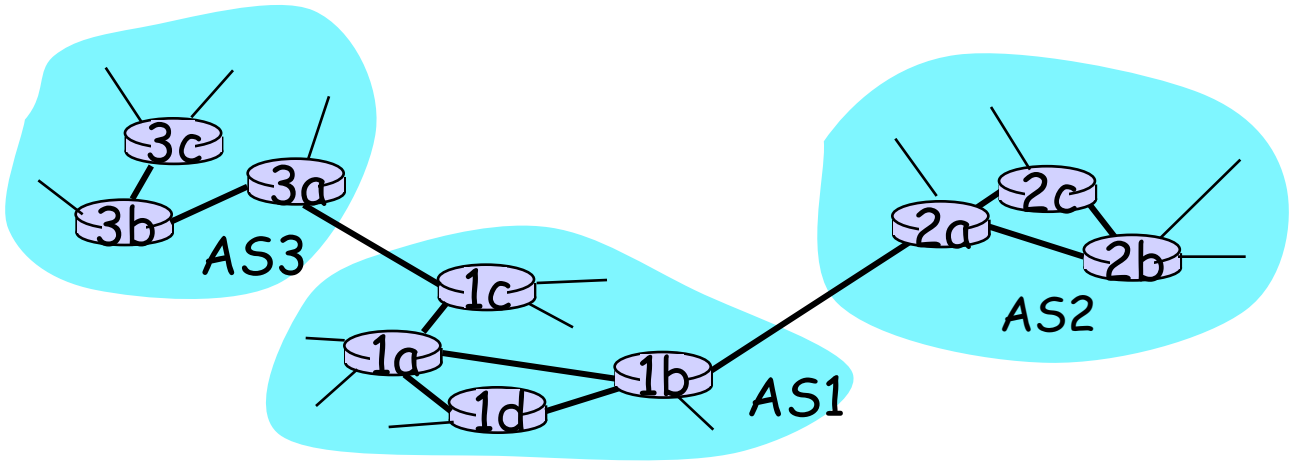
Tareas inter-AS

- supongamos que un router en AS1 recibe un datagrama destinado fuera de AS1:
  - el router debe encaminar el paquete a un "gateway", pero cual?

AS1 debe:

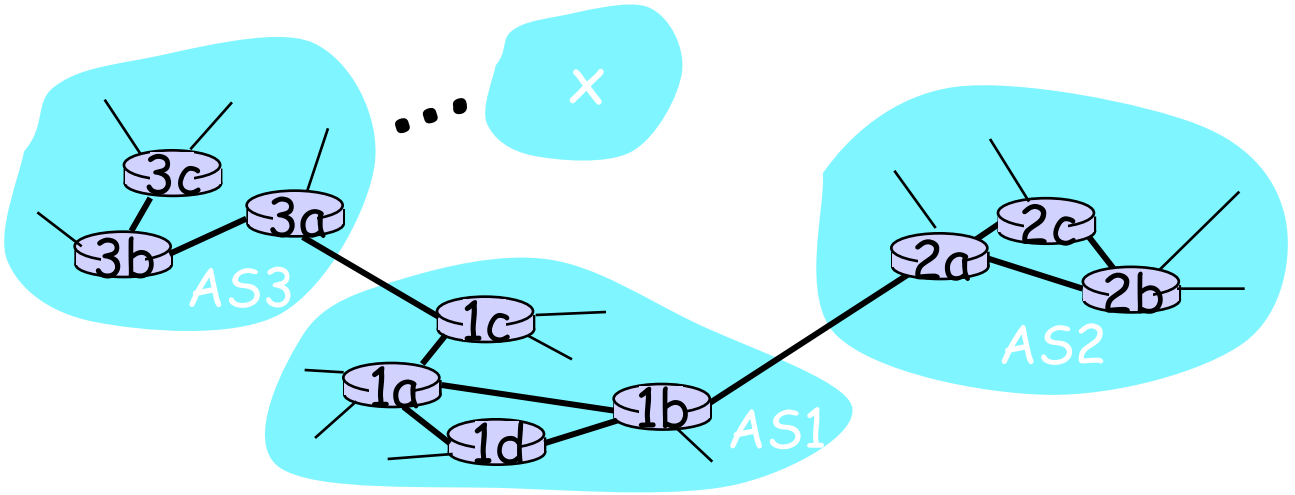
1. aprender que destinos son alcanzables por AS2, y cuales por AS3
2. Propagar esta información en AS1

Trabajo del routing inter-AS!



Ejemplo:
configuración de la tabla de forwarding en router 1d

- supongamos que AS1 sabe (via protocolo inter-AS) que la subred *x* es alcanzable via AS3 (gateway 1c) pero no via AS2.
- el protocolo inter-AS protocol propaga información de alcanzabilidad a los routers internos.
- Usando esta información, el router 1d determina que su interfaz *I* está en el camino de menor costo a 1c.
  - instala la entrada *(x,I)* en su tabla de forwarding

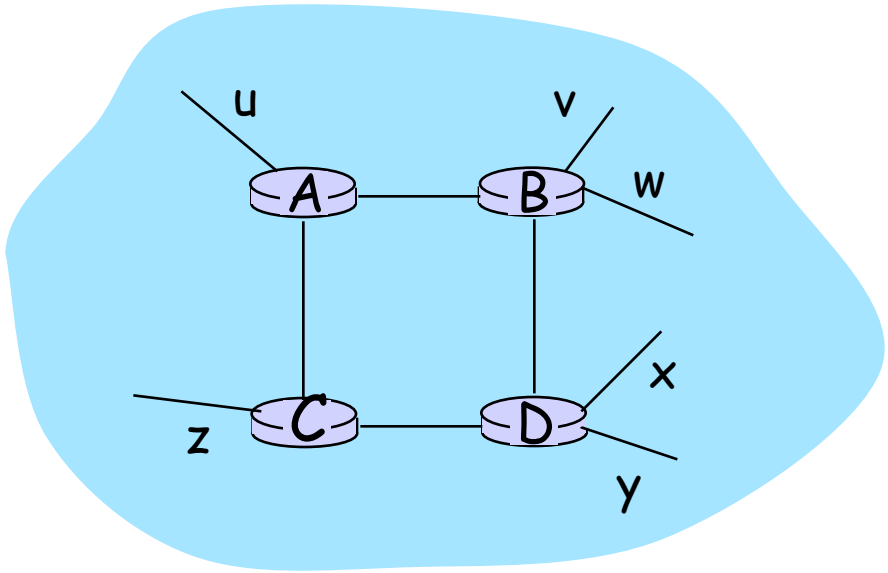


Routing intra-AS

- a.k.a. Interior Gateway Protocols (IGP)
- los más comunes:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (propietario de Cisco)

RIP (Routing Information Protocol)

- algoritmo distance vector
- incluido en la distribución de BSD-UNIX en 1982
- métrica de distance: # of hops (máx = 15 hops)



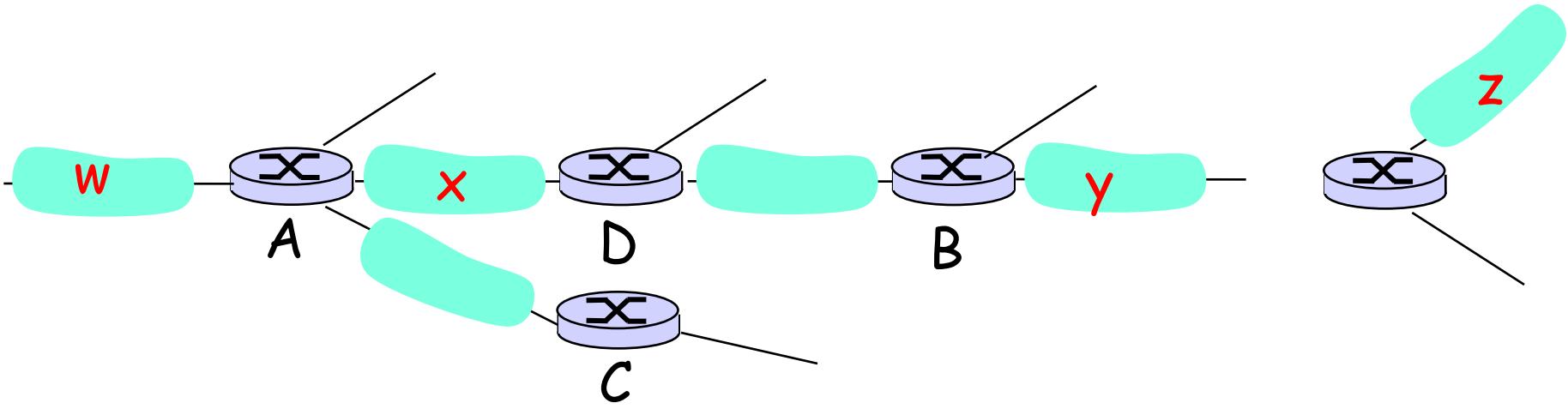
del router A a subredes:

destino	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP advertisements

- *distance vectors:* intercambiados entre vecinos cada 30 segs. via Response Message (también llamado advertisement)
- cada advertisement: lista de hasta 25 subredes destino dentro del AS

RIP: Ejemplo



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	....

Routing/Forwarding table in D

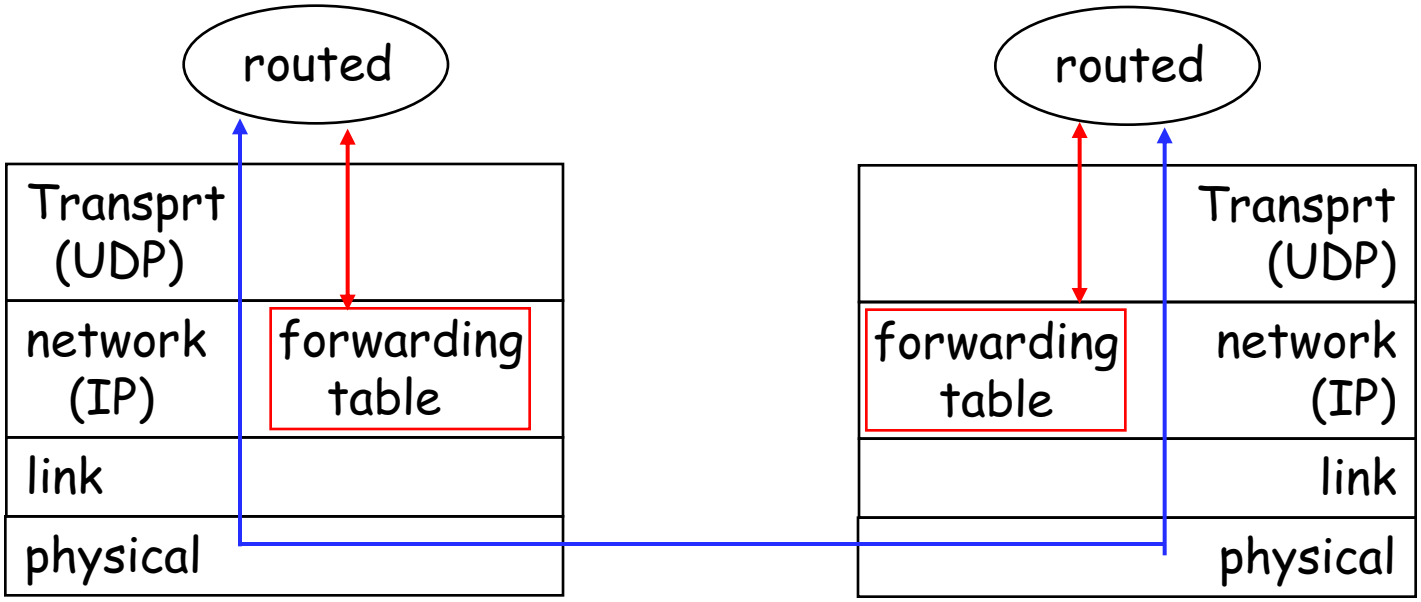
RIP: Link Failure & Recovery

Si no se recibe un advertisement pasados 180 segs. --> vecino/enlace es declarado "muerto"

- se invalidan las rutas via este vecino
- se envían nuevos advertisements a vecinos…
- …que a su vez envían nuevos advertisements (si hay cambios en las tablas)
- Fallo en enlace se propaga a toda la red, rápidamente (?)
- *poison reverse* usado para prevenir loops (ping-pong); distamcia infinita = 16 hops)

RIP: procesamiento de la tabla

- La tabla de enrutamiento de RIP es gestionada por un proceso de capa de aplicación, llamado route-d (daemon)
- los advertisements se envían en paquetes UDP



OSPF (Open Shortest Path First)

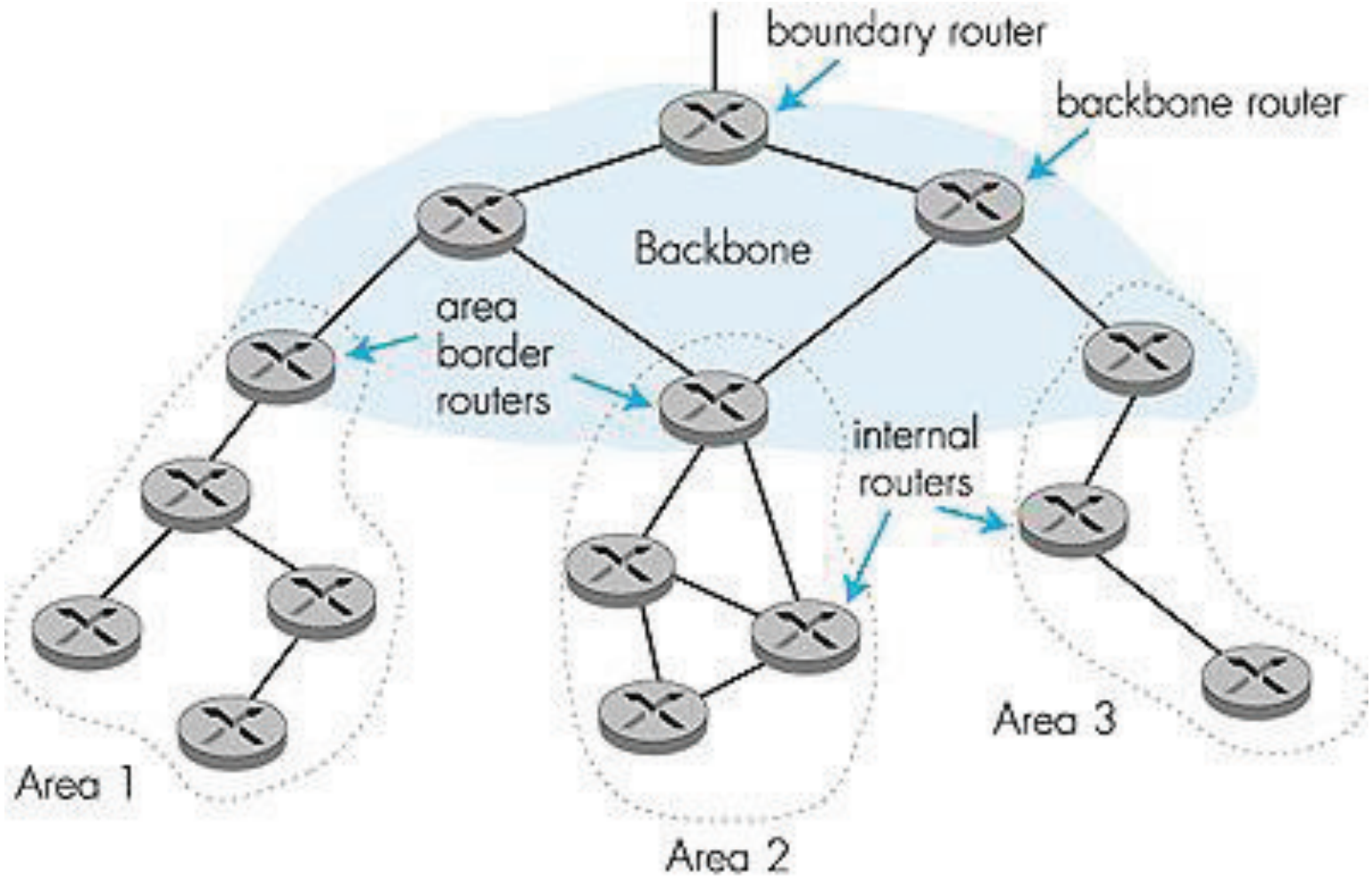
- "open": disponible públicamente
- usa algoritmo Link State
  - diseminación de paquetes LS
  - mapa de la topología en cada nodo
  - Cómputo de rutas usando el algoritmo de Dijkstra

- advertisement de OSPF transporta una entrada para cada router vecino
- los advertisements son diseminados a todo el AS (via flooding)
  - los mensajes OSPF son transportados directamente sobre IP (en lugar de TCP or UDP)

Carácterísticas "avanzadas" de OSPF (no en RIP)

- seguridad: todos los mensajes OSPF son autenticados (para prevenir intrusiones maliciosas)
- se admiten múltiples caminos de igual costo (solo uno en RIP)
- para cada enlace, métricas de costo diferentes según TOS (ej., el costo de un enlace satelital se configura "bjo" para best effort; "alto" para tiempo real)
- Soporte integrado uni y multicast:
  - Multicast OSPF (MOSPF) usa la misma base de datos de topología que OSPF
- OSPF jerárquico en dominios grandes.

OSPF Jerárquico



OSPF Jerárquico

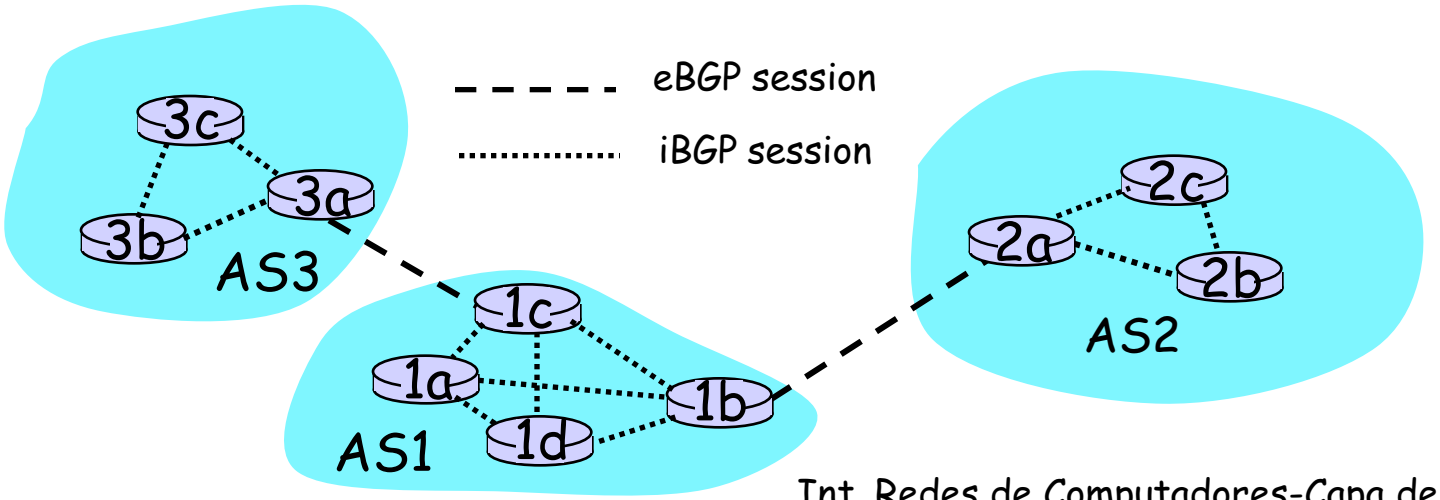
- jerarquía de dos niveles: área local, backbone.
  - Link-state advertisements solo en el área
  - cada nodo conoce la topología detallada del área, pero solo resúmenes de las subredes en otras áreas.
- *area border routers:* "sumarizan" distancias a redes en el área propia, y lo publican hacia los otros Area Border routers.
- *backbone routers:* OSPF limitado al backbone.
- *boundary routers:* conectan con otros ASs (gateways o routers de borde).

Inter-AS routing en Internet: BGP

- BGP (Border Gateway Protocol): estándar *de facto*
- BGP provee mecanismos para:
  1. Obtener información de alcanzabilidad de subredes de los ASs vecinos.
  2. Propaga información de alcanzabilidad a los routers internos del AS.
  3. Determina que rutas son "buenas" basadas en la información de alcanzabilidad y las políticas de enrutamiento.
- permite informar la alcanzabilidad de subredes al resto de Internet: *"aquí estoy"*

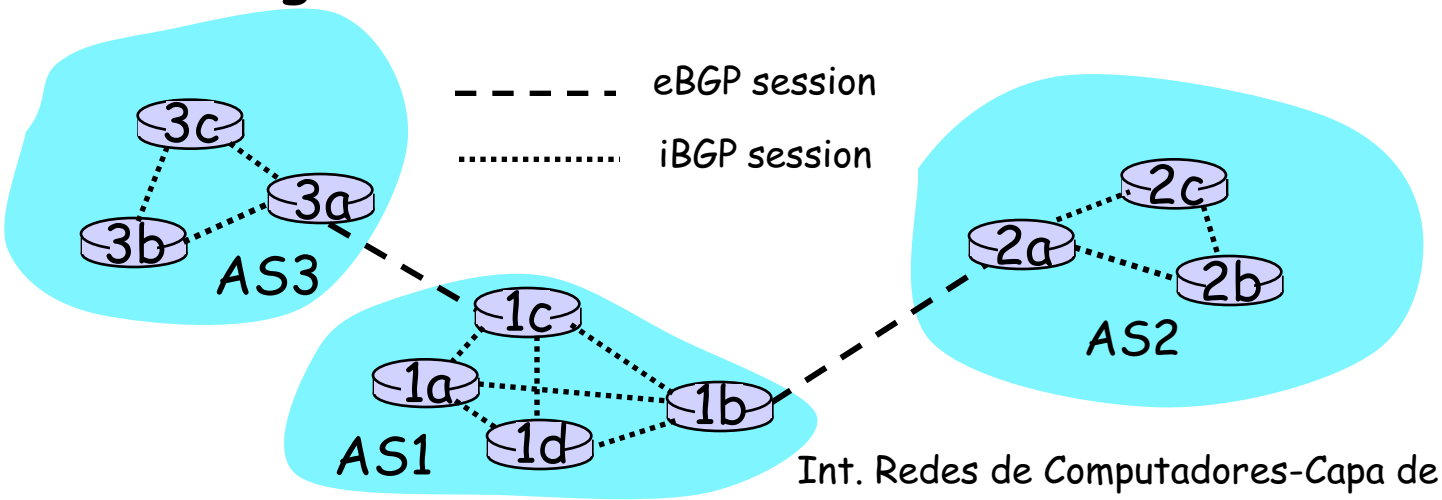
BGP: conceptos básicos

- pares de routers (BGP peers) intercambian información de routing sobre conexiones TCP semi-permanentes: sesiones BGP
  - Las sesiones BGP no se corresponden necesariamente con enlaces físicos.
- cuando AS2 publica un prefijo a AS1:
  - AS2 *"promete"* encaminar datagramas para ese prefijo.
  - AS2 puede agregar prefijos es sus publicaciones



Distribución de la información de alcanzabilidad

- AS3 envía la información de alcanzabilidad a AS1 usando una sesión eBGP entre 3a y 1c.
  - 1c puede usar iBGP para distribuir la información de prefijos a los routers en AS1
  - 1b puede re-publicar la información hacia AS2 usando la sesión eBGP 1b-2a
- Cuando un router aprende un prefijo nuevo, crea una entrada para este prefijo en la tabla de forwarding.



Path attributes & rutas BGP

- Las publicaciones de prefijos incluyen atributos BGP.
  - prefjo + atributos = "rutas"
- dos atributos importantes:
  - AS-PATH: contiene la lista de ASs que ha atravesado la publicación de un prefijo: ej., AS 67, AS 17
  - NEXT-HOP: indica el router específico en el próximo AS (pues puede haber múltiples enlaces entre ASs).
- Cuando un router de borde recibe una publicación, usa su "import policy" para aceptar/rechazar.

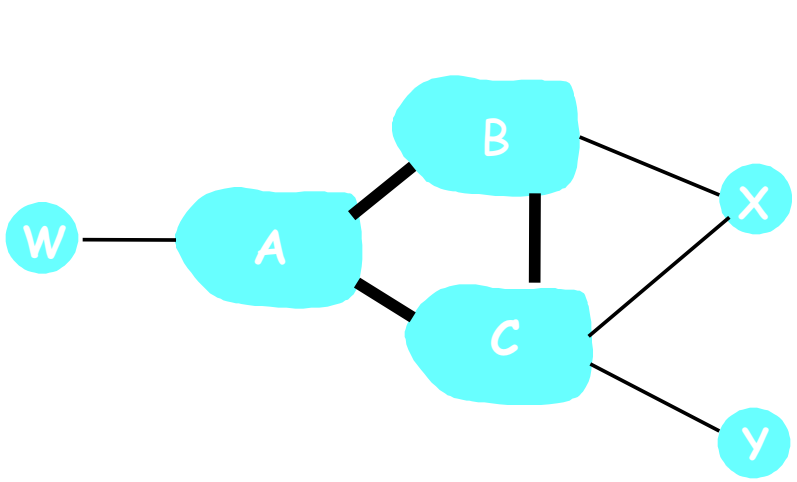
BGP route selection

- un router puede aprender más de una ruta para un prefijo dado: se necesita un proceso de selección.
- reglas de eliminación:
  1. atributo "local preference": política de decisión
  2. shortest AS-PATH
  3. closest NEXT-HOP router: hot potato routing
  4. criterios adicionales

Mensajes BGP

- los mensajes BGP se intercambian usando TCP.
- mensajes BGP:
  - OPEN: abre conexión TCP con "peer" y autentica al que envía
  - UPDATE: publica nuevos caminos (o da de baja otros)
  - KEEPALIVE: mantiene la conexión viva en ausencia de UPDATES; se usa también como ACK del OPEN
  - NOTIFICATION: reporta errores en mensaje previo; también se usa para cerrar conexión

BGP routing policy



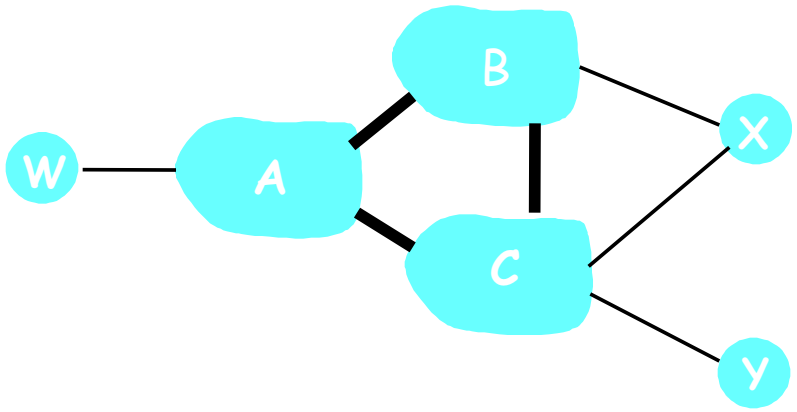
referencia:

red del proveedor

red del cliente

- A,B,C son redes de proveedores
- X,W,Y son clientes
- X es dual-homed: conectado a dos proveedores
  - X no permite enrutar desde B via X hacia C…
  - … luego X no va a publicar a B un ruta hacia C

BGP routing policy



referencia:

red del proveedor

cliente

- A publica camino AW a B
- B publica camino BAW a X
- debería B publicar camino BAW a C?
  - No! B no tiene "retorno" por enrutar CBAW dado que ni W ni C son sus clientes
  - B quiere forzar que C enrute hacia w via A
  - B quiere enrutar *solo* desde/hacia sus clientes!

Por qué Intra- e Inter-AS routing ?

Policy:

- Inter-AS: los administradores quieren controlar como se enruta su tráfico, y quien usa el AS como tránsito.
- Intra-AS: administración única, no se necesitan políticas

Escala:

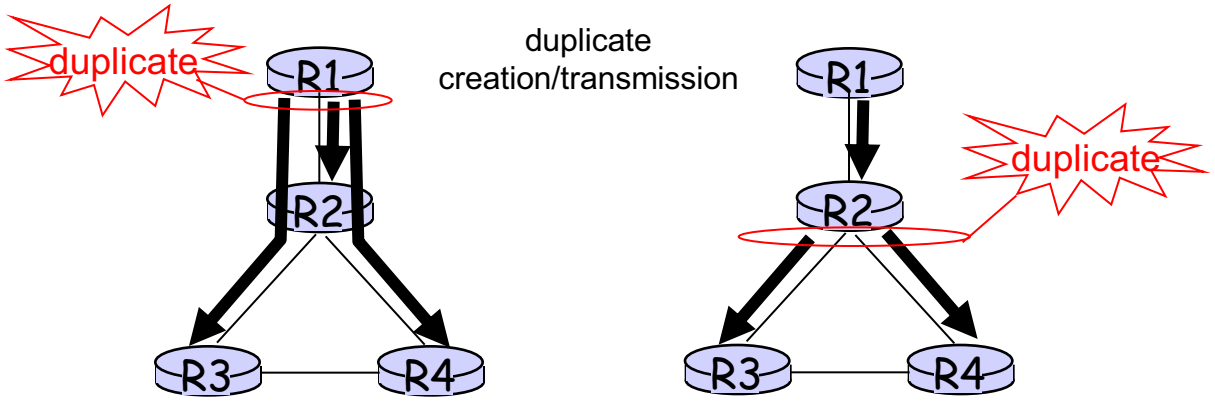
- enrutamiento jerárquico reduce el tamaño de las tablas y de la información de actualización de enrutamiento

Performance:

- Intra-AS: enfocado en performance
- Inter-AS: políticas son más importantes que performance

Broadcast Routing

- entrega de paquetes desde la fuente a todos los nodos
- duplicación en la fuente es ineficiente:



Duplicación en la fuente

Duplicación en la red

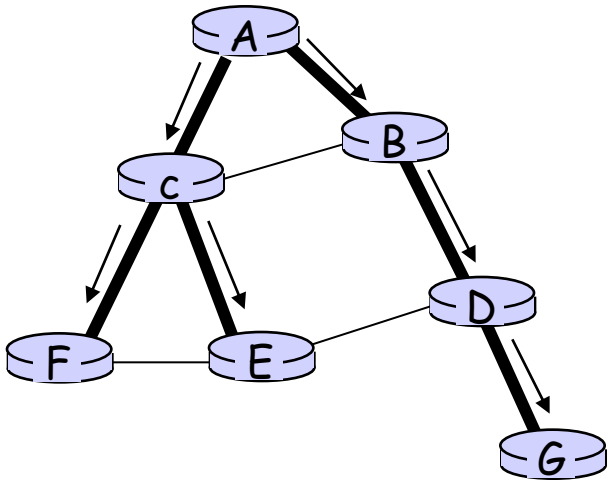
- duplicación en la fuente: como determinar la dirección de los receptores?
  - registro?

Duplicación en la red

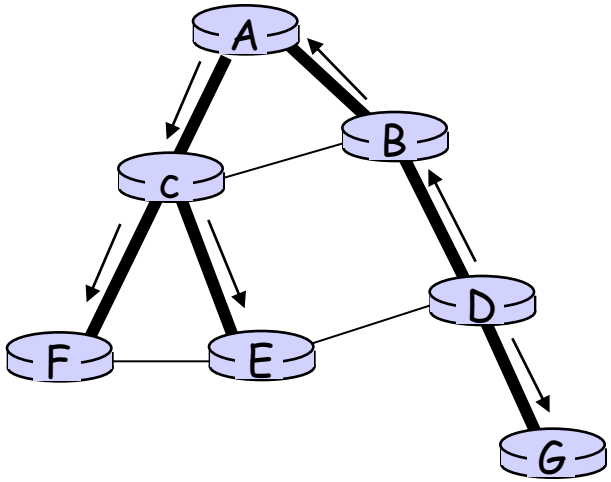
- flooding: cuando un nodo recibe un paquete, envía copias a todos sus vecinos
  - Problemas: ciclos & "tormenta" de broadcasts
- flooding controlado: el nodo solo hace broadcast de un paquete si no lo ha enviado antes
  - el nodo debe llevar la cuenta de los paquetes enviados recientemente
  - o "reverse path forwarding" (RPF): solo envía un paquete si llegó por el camino más corto entre el nodo y la fuente
- spanning tree
  - ningún nodo recibe paquetes redundantes

Spanning Tree

- primero hay que contruir el spanning tree
- los nodos envían copias solamente sobre el spanning tree



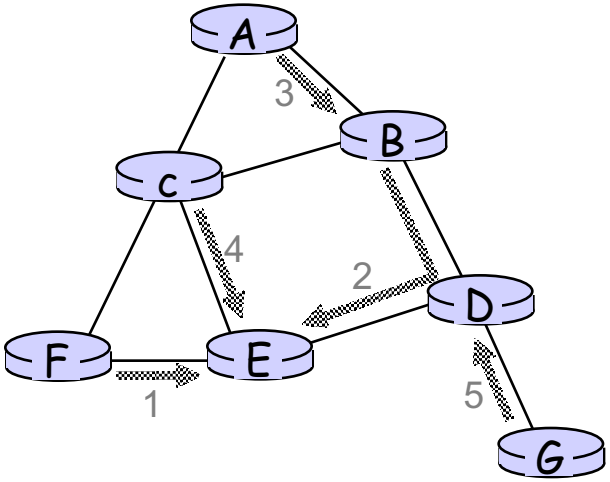
(a) Broadcast iniciado en A



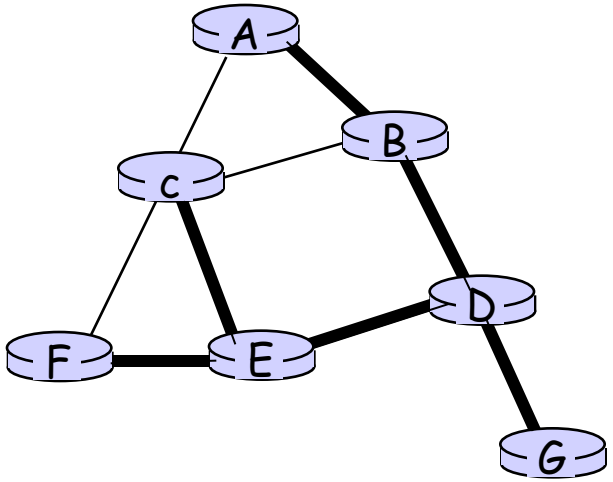
(b) Broadcast initiado en D

Spanning Tree: creación

- nodo central o raíz
- cada nodo envía un mensaje unicast al nodo central para unirse al árbol
  - el mensaje es reenviado hasta que llega a un nodo que pertenece al spanning tree



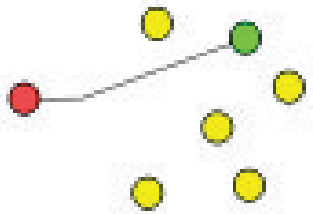
(a) Construction paso a paso del spanning tree



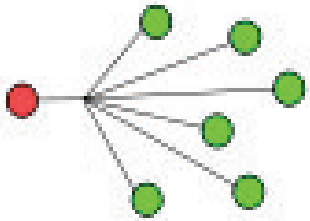
(b) Spanning tree construido

Multicast

UNICAST: UN EMISOR, UN RECEPTOR.

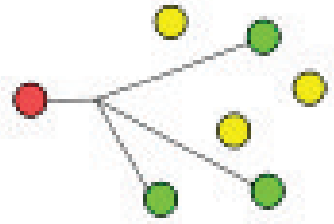


BROADCAST: UN EMISOR, TODOS LOS RECEPTORES.



ANYCAST: VARIOS EMISORES, UN RECEPTOR (*).

MULTICAST: VARIOS EMISORES, VARIOS RECEPTORES VOLUNTARIOS.



Grupos de multicast

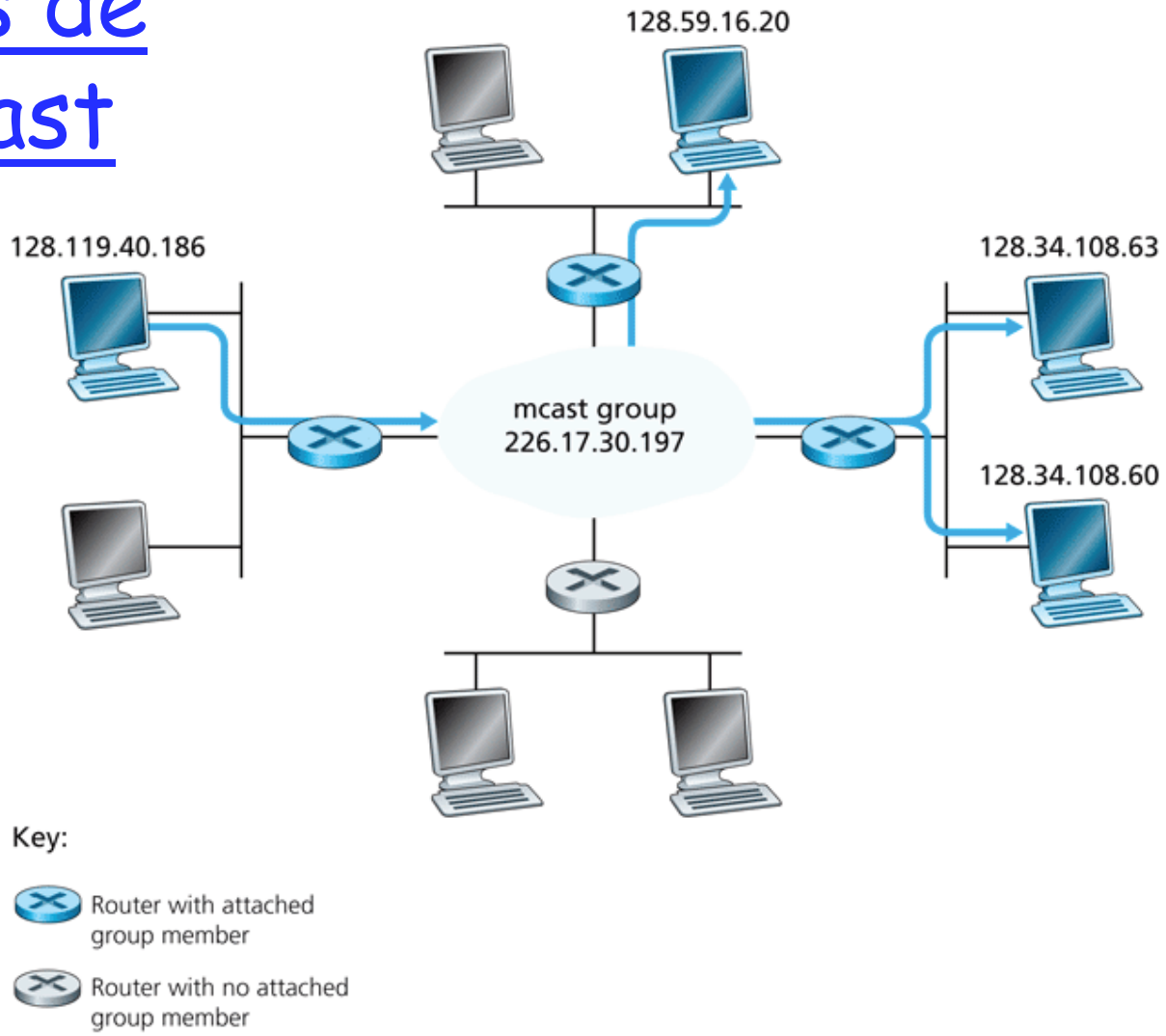
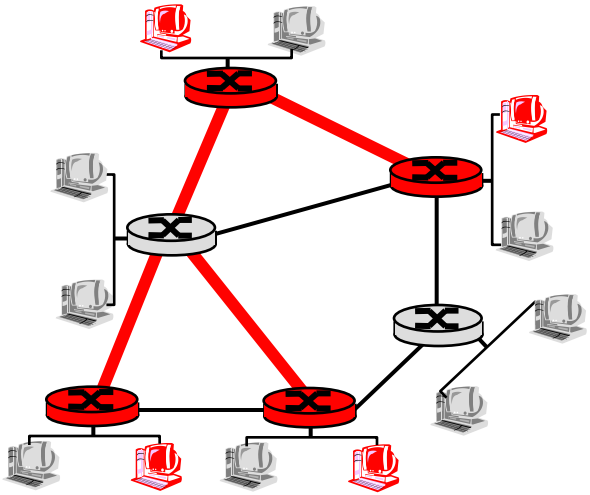


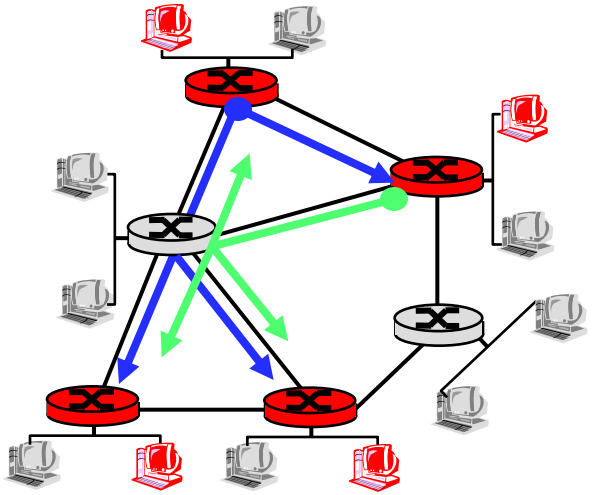
Figure 4.48 ♦ The multicast group: A datagram addressed to the group is delivered to all members of the multicast group.

Multicast Routing: el problema

- ***Objetivo:*** encontrar un árbol (o árboles) que conecta routers que tienen miembros de grupos de mcast
  - *árbol:* no se usan todos los caminos
  - *source-based:* árboles diferentes desde cada fuente a receptores
  - *shared-tree:* todos los miembros del grupo usan el mismo árbol



Shared tree



Source-based trees

Como se construyen los árboles de multicast?

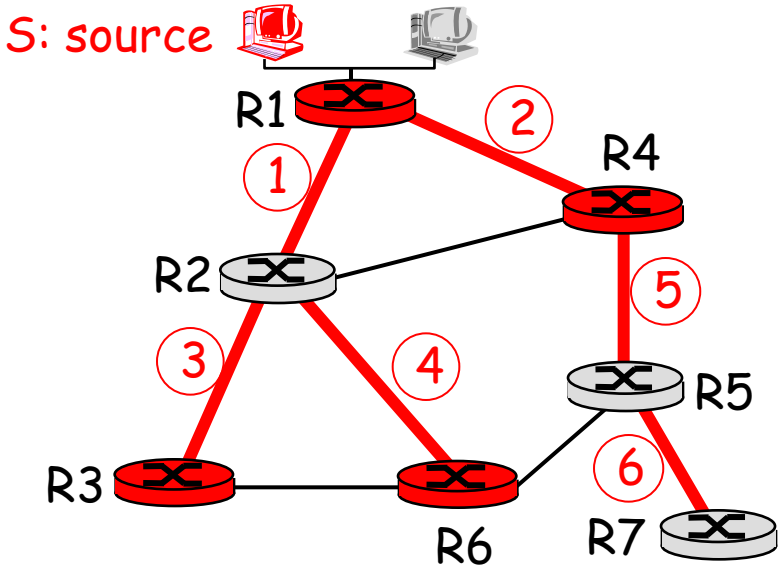
Posibilidades:

- source-based tree: un árbol por fuente
  - shortest path trees
  - reverse path forwarding
- group-shared tree: el grupo usa un único árbol
  - minimal spanning (Steiner)
  - center-based trees

…primero vamos a ver los enfoques básicos, y algunos protocolos que los implementan

Shortest Path Tree

- mcast forwarding tree: árbol de caminos más cortos desde la fuente a todos los receptores
  - Algoritmo de Dijkstra



REFERENCIA

router con miembros del grupo conectados

router sin miembros del grupo conectados

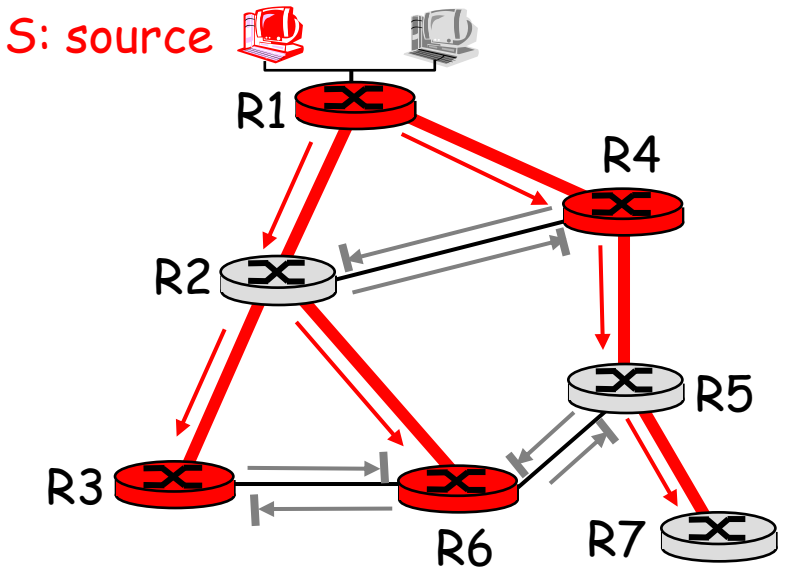
i — Enlace usado para forwarding, i indica el orden en que fue agregado por el algoritmo

Reverse Path Forwarding

- Se basa en el conocimiento que tiene el router del "unicast shortest path" desde si mismo a la fuente
- cada router se comporta de forma simple:

> ***if*** (datagrama mcast recibido en enlace entrante en el "shortest path" hacia el centro)
>
> ***then*** "flooding" del datagram en los enlaces de salida
>
> ***else*** ignorar datagrama

Reverse Path Forwarding: ejemplo



REFERENCIA

router con miembros del grupo conectados

router sin miembros del grupo conectados

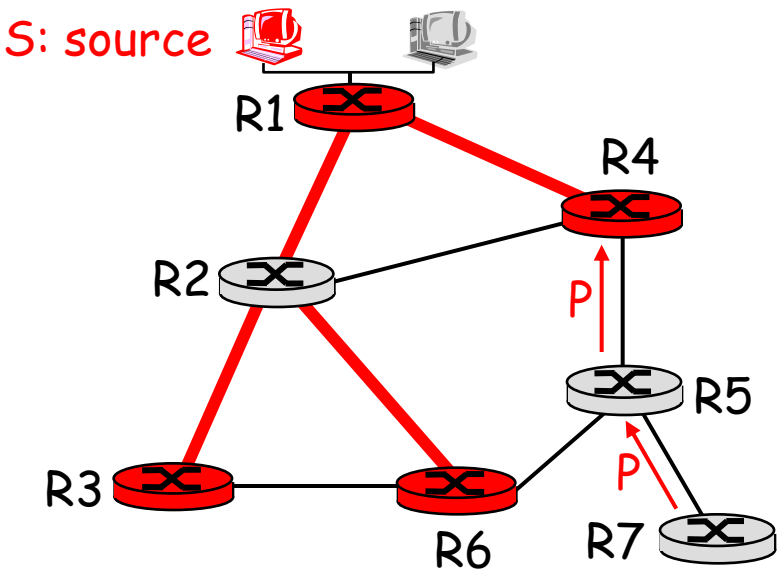
forwarding del datagrama

NO forwarding del datagrama

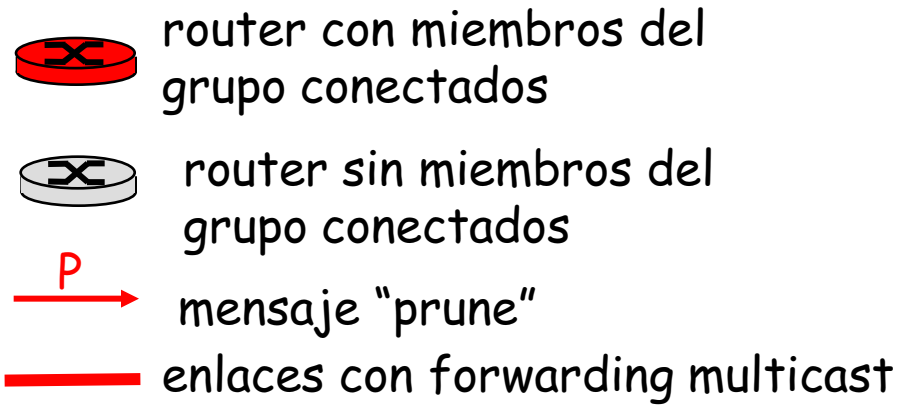
- el resultado es un "*reverse* SPT" específico para la fuente
  - puede ser una mala opción con enlaces asimétricos

Reverse Path Forwarding: pruning

- el árbol de forwarding contiene sub-árboles sin miembros del grupo de mcast conectados
  - no es necesario reenviar datagramas en estos sub-árboles
  - Los routers que no tienen miembros conectados envían mensaje "prune" (poda) "hacia atrás"



REFERENCIA



Shared-Tree: Steiner Tree

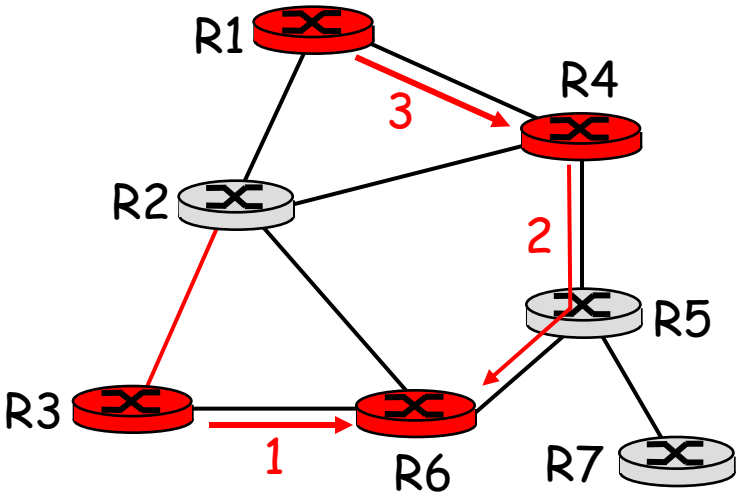
- Steiner Tree: árbol de costo mínimo que conecta todos los routers con miembros del grupo conectados
- problema NP-completo
  - existen buenas heurísticas para atacarlo
- no se usa en la práctica:
  - complejidad computational
  - se necesita información de toda la red
  - monolítico: debe recalcularse cada vez que un router hace un "join"/"leave"

Center-based trees

- árbol de entrega único compartido
- un router identificado como *"centro"* del árbol
- proceso de unión (join):
  - Los routers de borde envían mensajes unicast *join* destinados al router central
  - el mensaje *join* es procesado por los routers intermedios y reenviados hacia el centro
  - el mensaje *join* se une a una rama existente, o llega al centro
  - el camino recorrido por el mensaje *join* se convierte en una nueva rama

Center-based trees: un ejemplo

Supongamos que R6 se elige como centro:



REFERENCIA

router con miembros del grupo conectados

router sin miembros del grupo conectados

1 recorrido de los mensajes *join*

Internet Group Management Protocol: IGMP

- hemos visto como se arman los árboles de distribución...
- ...pero todavía no sabemos como hace un host para unirse a un grupo de multicast
- IGMP: protocolo usado por los routers locales (los default gateways) y los hosts

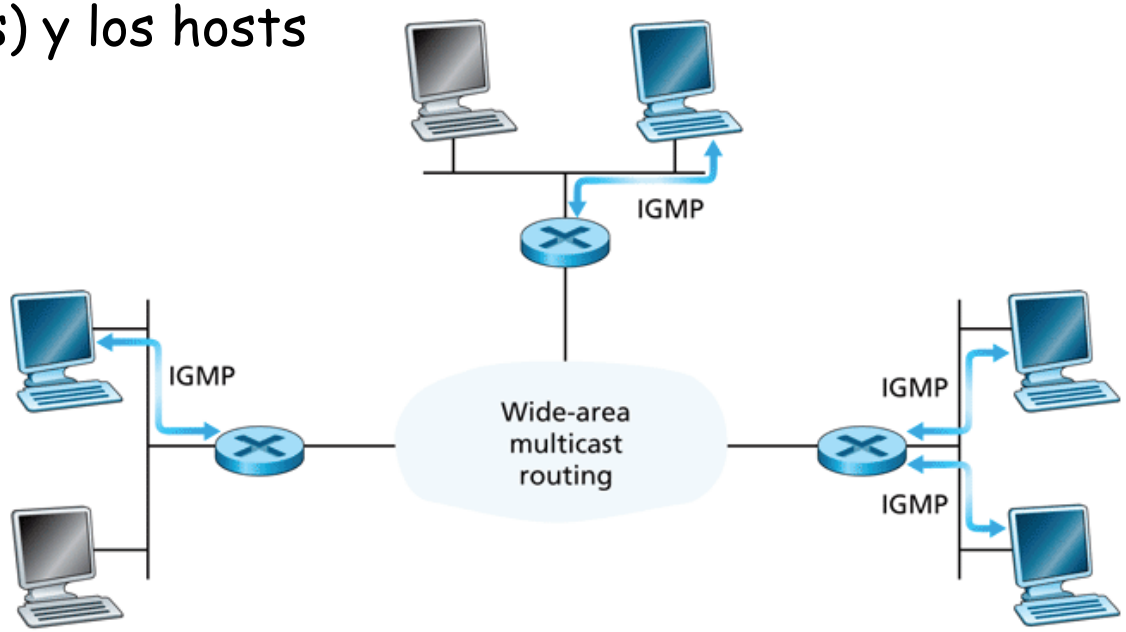


Figure 4.49 ♦ The two components of network-layer multicast in the Internet: IGMP and multicast routing protocols

IGMP

- mensajes:
  - query: desde el router a los hosts
  - membership report: desde los hosts a los routers
  - leave: desde los hosts a los routers
- el router puede hacer queries genéricos ("todos los grupos") o específicos
- los hosts pueden enviar mensajes de asociación sin esperar ser interrogados (no solicitados)
- soft state: el router mantiene una tabla de los hosts que pertenecen a cada grupo enviando queries periódicos y recibiendo las respuestas

- IGMP: protocolo de control análogo a ICMP para unicast

Internet Multicasting Routing: DVMRP

- DVMRP: distance vector multicast routing protocol, RFC1075
- *flood & prune:* reverse path forwarding, source-based tree
  - árbol RPF basado en las tablas de routing de DVMRP, construida por intercambio de mensajes nativos (no se basa en unicast)
  - el datagrama inicial se envía por flooding al grupo de mcast usando RPF
  - los routers que no participan del grupo envían mensajes *prune* hacia "arriba"

DVMRP

- *soft state:* los routers DVMRP "se olvidan" periodicamente (1 min.) que las ramas que están "*pruned*":
  - se vuelven a enviar flujos mcast por esas ramas
  - los routers "downstream" (hacia abajo) tienen que volver a mandar prune o seguirán recibiendo el flujo
- los routers pueden reconectarse
  - siguiendo los *join* de IGMP en las hojas (redes locales)
- "de la vida real"
  - usualmente implementado en routers comerciales
  - usado en Mbone (también MOSPF)