Tecnología Digital IV: Redes de Computadoras

Clase 15: Nivel de Red - Parte 3

Lucio Santi & Emmanuel Iarussi

Licenciatura en Tecnología Digital
Universidad Torcuato Di Tella

15 de mayo de 2025

Agenda

- Vista general del nivel de red
  - Plano de datos
  - Plano de control
- Routers
  - Puertos de *input* y *output*
  - Switching
- El Protocolo de Internet (**IP**)
  - Datagrama
  - Direccionamiento
  - NAT
  - IPv6
- *Software Defined Networking* (SDN)

Obtención de direcciones IP

1. ¿Cómo hace un host para obtener una dirección IP dentro de su red?

1. ¿Cómo obtener una dirección IP para una red en Internet?

Obtención de IPs dentro de una red:

- Configuración manual por un administrador en un archivo (e.g., `/etc/network/interfaces` en Ubuntu pre 20.04/Debian)
- DHCP (*Dynamic Host Configuration Protocol*)
  - Obtención **dinámica** de dirección IP desde un servidor

DHCP: *Dynamic Host Configuration Protocol*

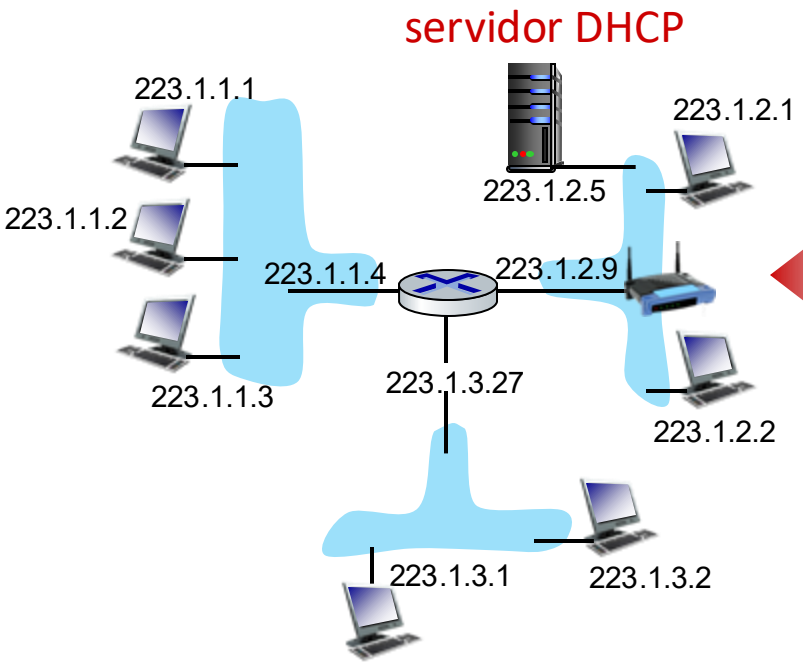
Objetivo: permitir que los hosts consigan direcciones IP dinámicamente al “unirse” a la red

- “Préstamo” de direcciones (renovables)
- Direcciones reutilizables (sólo tomadas mientras los hosts están conectados)
- Soporte para usuarios móviles que entran y salen de la red

Vista general del protocolo:

- El host hace *broadcast* de un mensaje *DHCP discover*
- El servidor DHCP responde con un *DHCP offer*
- El host solicita una dirección IP con un *DHCP request*
- El servidor DHCP envía la dirección con un *DHCP ACK*

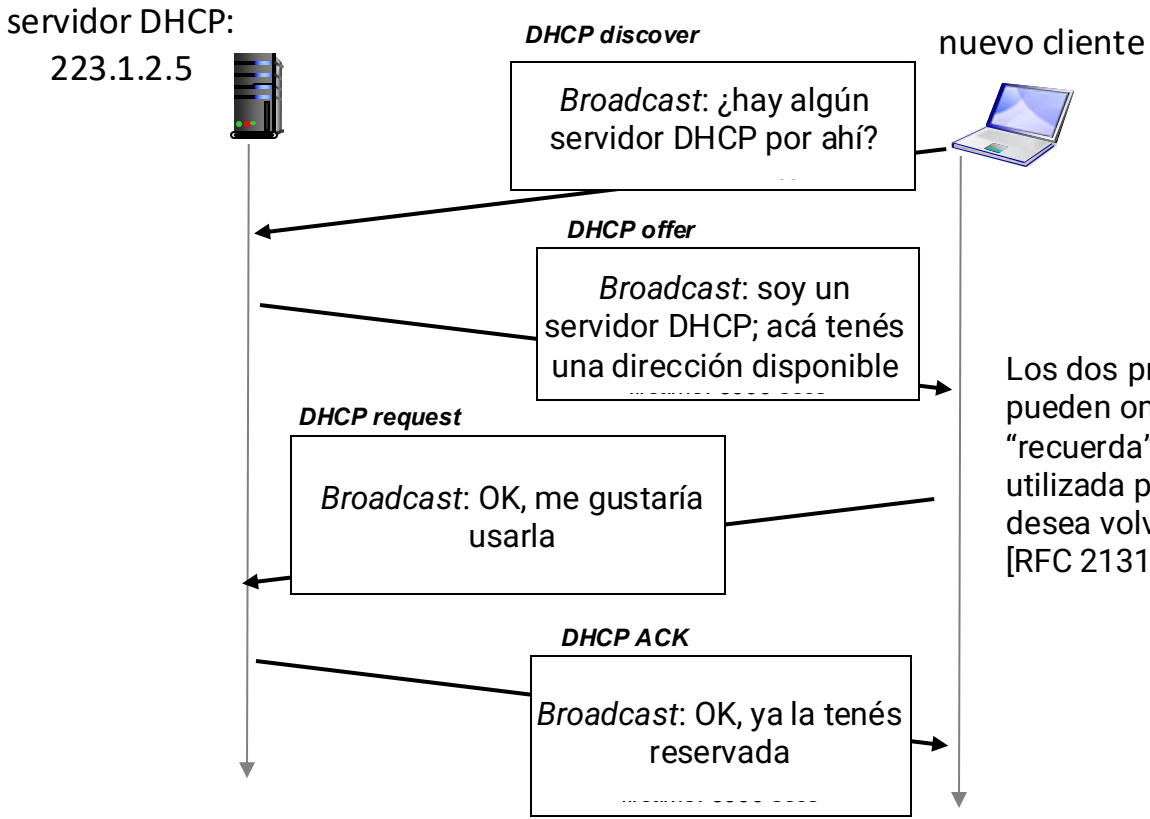
DHCP: *Dynamic Host Configuration Protocol*



Por lo general, el servidor DHCP está embebido en el router, sirviendo a todas las *subnets* a las cuales se conecta el router

Al llegar un nuevo cliente, inicia el intercambio de mensajes para obtener una dirección IP en la red

DHCP: *Dynamic Host Configuration Protocol*



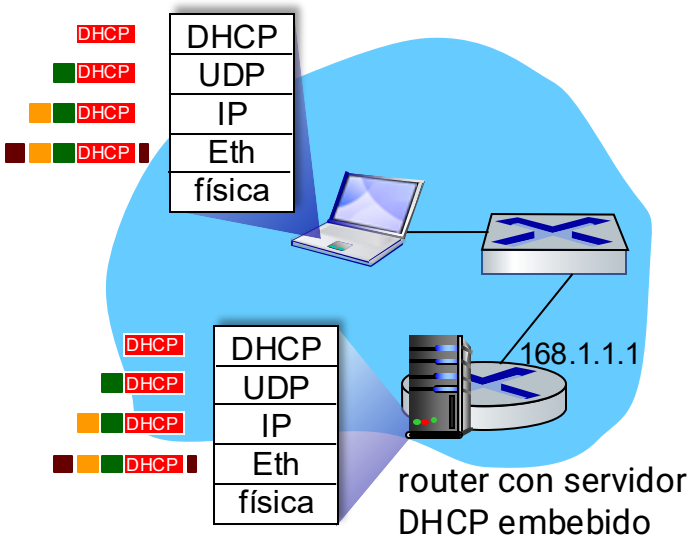
Los dos primeros pasos pueden omitirse si el cliente "recuerda" una dirección utilizada previamente y desea volver a usarla [RFC 2131]

DHCP: *Dynamic Host Configuration Protocol*

DHCP también permite que los hosts reciban, además de la dirección IP,

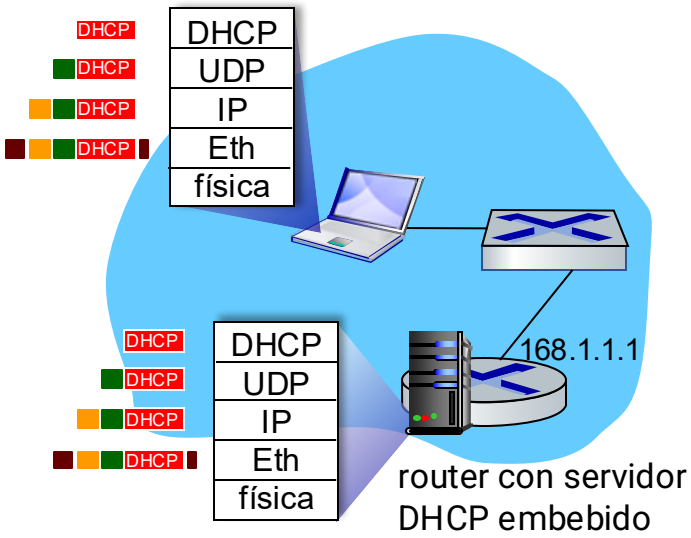
- La dirección del router *first-hop* (el *default gateway*)
- El nombre y la IP del servidor DNS local
- La máscara de red (indicando la porción de *subnet* en la dirección)

DHCP: encapsulamiento y demultiplexación



- La notebook usa DHCP para obtener una dirección IP, el default gateway y el servidor DNS
- El mensaje *DHCP Request* se encapsula en UDP; éste se encapsula en IP y este último en Ethernet
- El *frame* Ethernet se transmite a la LAN en *broadcast* (dirección física `FF:FF:FF:FF:FF:FF`) y es recibido por el router con el servidor DHCP
- En el router, Ethernet demultiplexa a IP, que a su vez demultiplexa a UDP, que a su vez demultiplexa a DHCP

DHCP: encapsulamiento y demultiplexación



- El servidor DHCP genera un mensaje DHCP ACK con la dirección IP asignada, la dirección del default gateway y la dirección del servidor DNS
- La respuesta DHCP, apropiadamente encapsulada, viaja hacia el cliente donde es demultiplexada
- El cliente recibe los datos solicitados y ya está en condiciones de comunicarse con hosts remotos

Demo!

- Utilizar **dhclient** (Linux) para renovar la dirección IP que tenemos asignada
- Inspeccionar el intercambio de paquetes en **Wireshark**:
  - ¿Cuál es la dirección IP del servidor DHCP?
  - ¿Qué dirección nos asignó?
  - ¿Cuánto tiempo dura el préstamo?
  - ¿Cuál/es servidor/es local/es DNS nos envió?

Obtención de direcciones de red

Los bloques de direcciones pueden obtenerse a través del ISP

- Tienen reservado un bloque más grande de direcciones

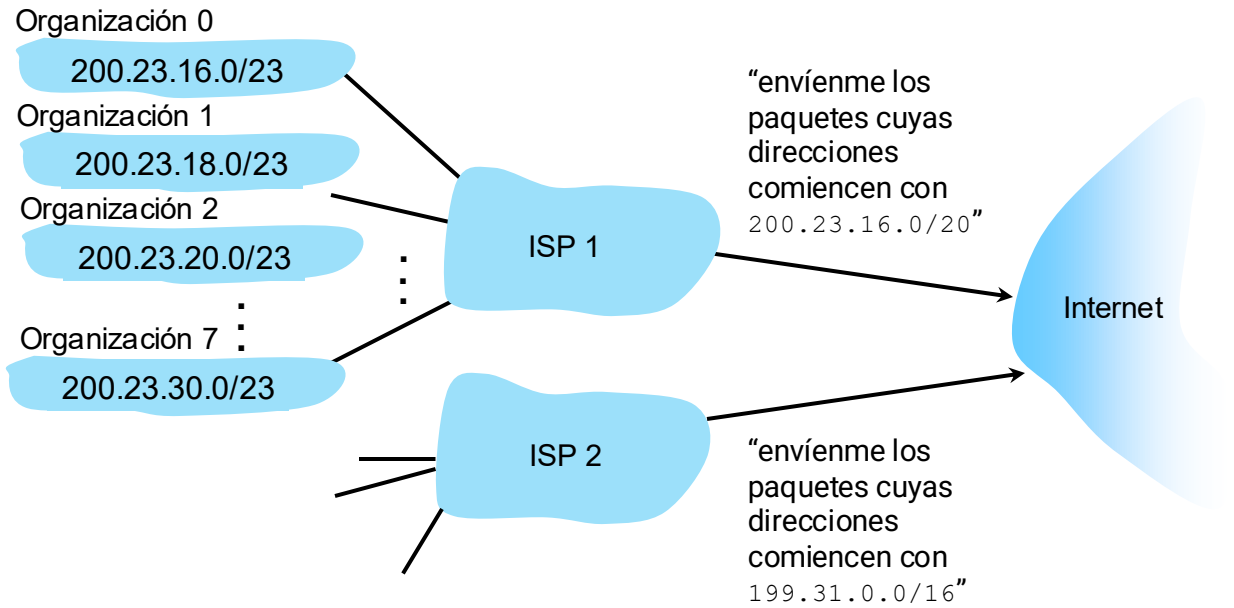
Bloque del ISP	11001000 00010111 00010000	00000000	200.23.16.0/**20**

El ISP puede e.g. particionar su espacio de direcciones en ocho bloques:

Bloque 0	11001000 00010111 00010000	00000000	200.23.16.0/**23**
Bloque 1	11001000 00010111 00010010	00000000	200.23.18.0/**23**
Bloque 2	11001000 00010111 00010100	00000000	200.23.20.0/**23**
…	…..	….	….
Bloque 7	11001000 00010111 00011110	00000000	200.23.30.0/**23**

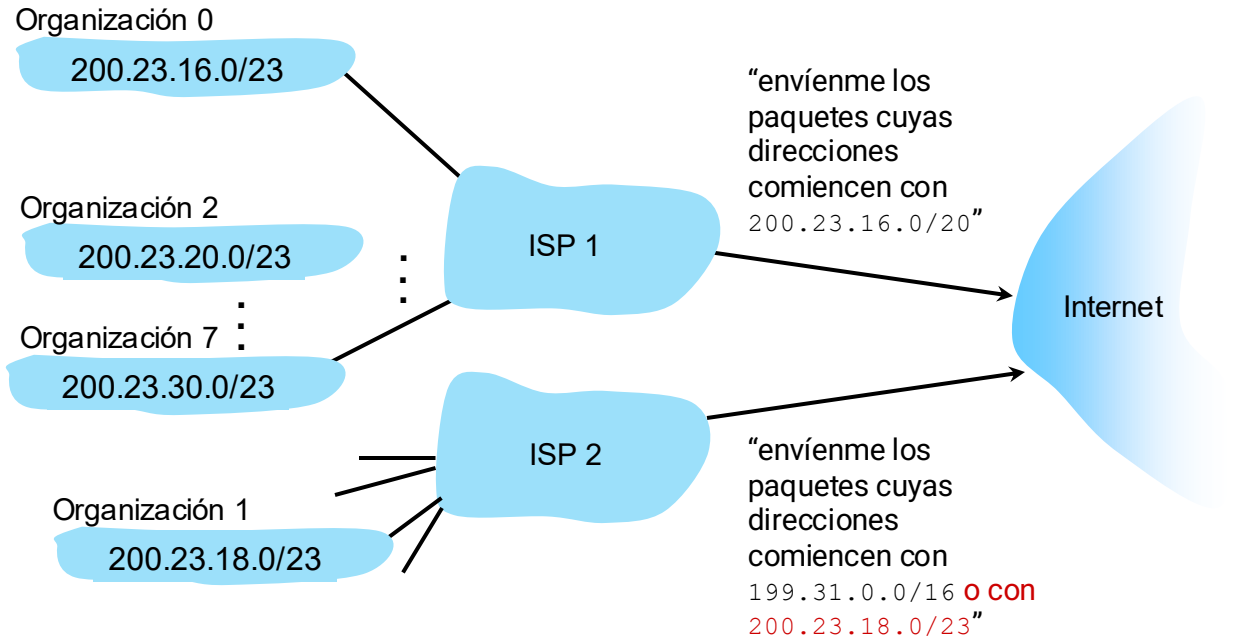
Direccionamiento jerárquico: agregación

Este **direccionamiento jerárquico** permite anunciar y manejar rutas de manera eficiente



Direccionamiento jerárquico

- Si e.g. la organización 1 se mueve de ISP, su espacio de direcciones **no cambia**: el ISP 2 anuncia también dicho bloque al exterior
- El mecanismo de *longest prefix matching* permite encaminar los paquetes



Direccionamiento IP

¿Cómo hace un ISP para obtener un bloque de direcciones?

ICANN: *Internet Corporation for Assigned Names and Numbers*

- Reserva direcciones IP a través de 5 *registros regionales* (RIRs) que a su vez reservan a registros locales (e.g. LACNIC en América Latina y Caribe)
- Administra la zona *root* de DNS, incluyendo delegación de los TLDs (e.g. .com o .edu)

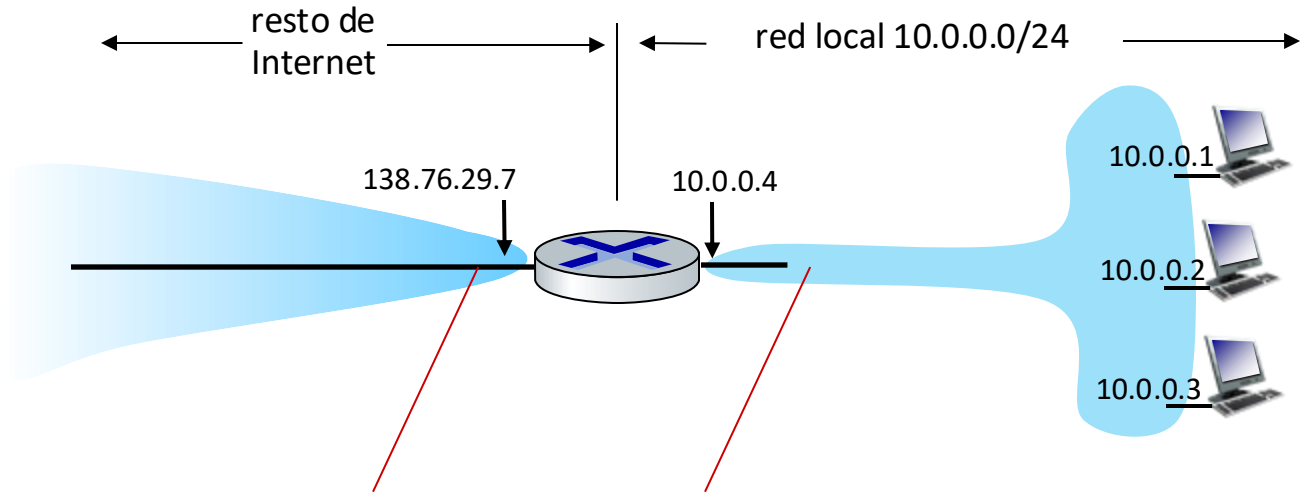
¿Hay suficientes direcciones IP de 32 bits?

- ICANN reservó a los RIRs el último bloque de direcciones IPv4 en 2011
- La técnica de **NAT** colabora para mitigar el agotamiento del espacio de direcciones IPv4
- **IPv6** tiene un espacio de direcciones de **128 bits**

NAT e IPv6

NAT: *Network Address Translation*

NAT: todos los dispositivos en la red local comparten una **única dirección IPv4** pública (i.e. visible desde el exterior)



todo datagrama saliente tiene la misma dirección IP origen: 138.76.29.7 (se diferencian por el puerto origen)

los datagramas con origen o destino en la red tienen dirección 10.0.0.0/24 como origen o destino (de la forma usual)

NAT: *Network Address Translation*

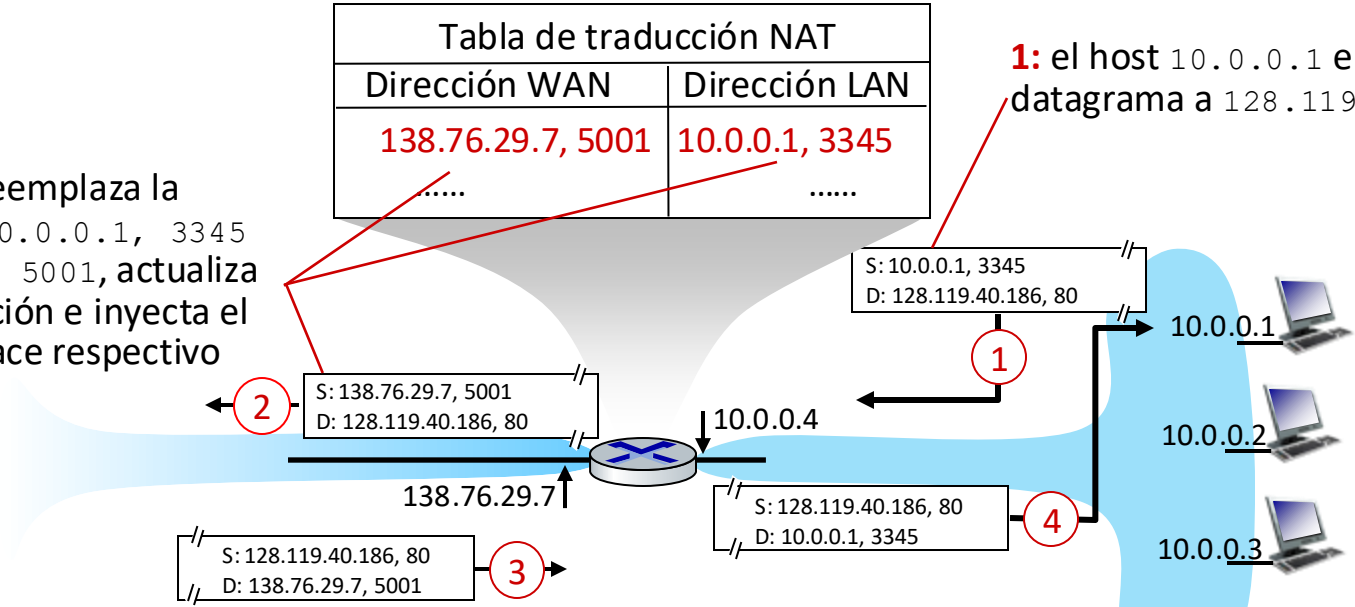
- Todos los dispositivos en una red local tienen direcciones de 32 bits en un espacio de direcciones **privado** (RFC 1918)
  - Prefijos `10.0.0.0/8`, `172.16.0.0/12` o `192.168.0.0/16`
  - Sólo pueden ser utilizadas en una red local
- Ventajas:
  - Sólo una dirección IP asignada por el ISP
  - Posibilidad de cambiar direcciones de hosts en la red sin notificar al mundo exterior
  - Posibilidad de cambiar de ISP sin cambiar las direcciones de los dispositivos
  - Seguridad: los dispositivos en la red no son directamente visibles/direccionables

NAT: *Network Address Translation*

Implementación: el router NAT debe (transparentemente):

- Reemplazar el par (*dirección IP origen*, *puerto origen*) de cada datagrama saliente por el par **(*dirección IP pública*, *nuevo puerto*)**
  - Los hosts remotos responden utilizando este último par como destino de sus segmentos
- Almacenar (en una tabla de traducción) los mapeos de (*dirección IP origen*, *puerto origen*) a (*dirección IP pública*, *nuevo puerto*)
- Hacer el reemplazo inverso en la dirección y el puerto destino de cada paquete entrante

NAT: *Network Address Translation*



NAT: *Network Address Translation*

- NAT tuvo sus **controversias**:
  - Los routers no deberían manipular información de niveles superiores (puertos)
  - El problema de la escasez de direcciones IP debería resolverse con IPv6
  - Complicaciones al tener servidores detrás de un router NAT
- No obstante esto, NAT llegó para quedarse
  - Ampliamente utilizado en redes domésticas e institucionales y en redes celulares 4G/5G (CGNAT; RFC 6598)

Demo!

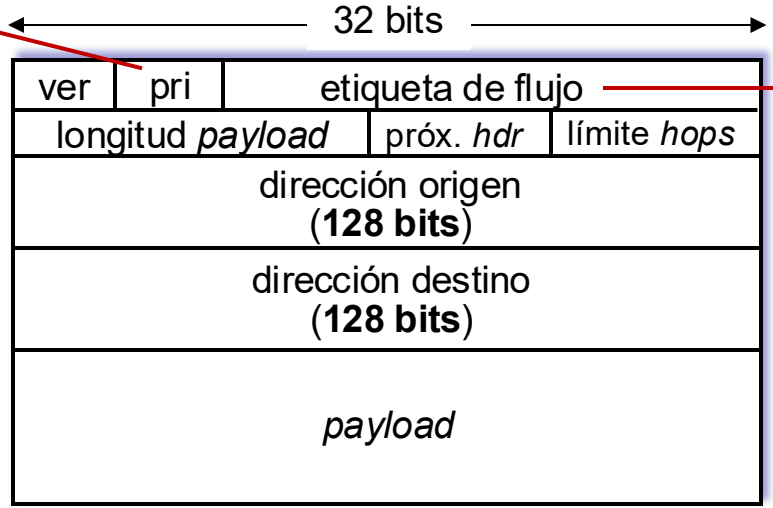
- Veamos cómo funciona el mecanismo de NAT en máquinas virtuales
- Vamos a configurar el adaptador de red de una VM para que utilice NAT y luego ejecutaremos algunas queries DNS en dicha VM
- Observar los paquetes con **Wireshark** corriendo fuera de la VM:
  - ¿Cuál es la dirección IP de los datagramas recibidos?
  - ¿Cuál dirección IP tiene asignada la máquina virtual?
  - ¿Dónde se está realizando la traducción?

IPv6: motivación

- Motivación inicial: escasez del espacio de direcciones de 32 bits de IPv4
- Otras motivaciones:
  - Mejorar la velocidad de procesamiento y *forwarding* en routers (*header* de longitud fija de 40 bytes)
  - Permitir otro tipo de manejo de “flujos” en el nivel de red

IPv6: formato del datagrama

prioridad entre datagramas de un flujo o para ciertas apps (símil ToS IPv4)



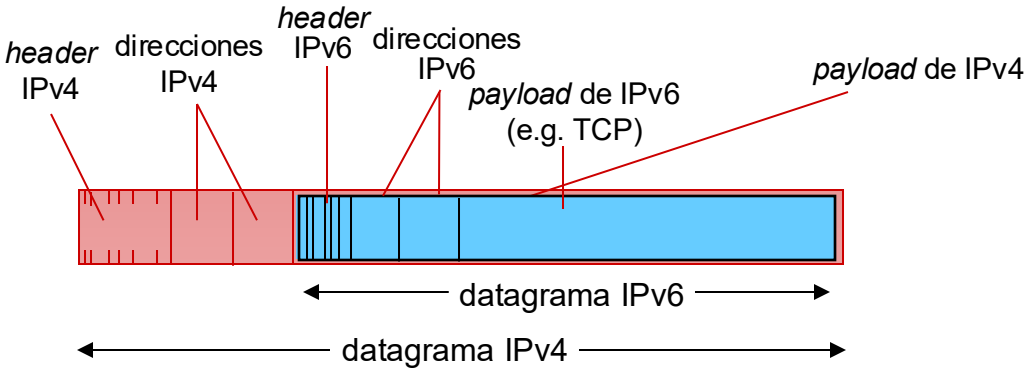
identificación de datagramas en el mismo "flujo" (concepto no muy bien definido; no se utiliza en la práctica: RFC 6294)

Qué falta (respecto de IPv4):

- No está el *checksum* (para optimizar el procesamiento en los routers)
- Sin fragmentación (operación costosa; sólo permitida en origen/destino)
- Sin opciones (disponibles como "próximo *header*")

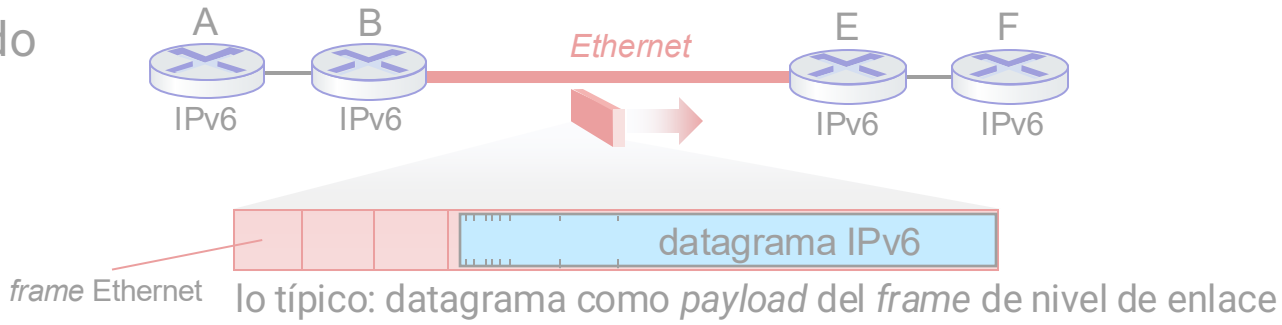
Transición IPv4 → IPv6

- No es posible actualizar todos los routers en simultáneo
  - **Miles de millones** de dispositivos
- *Tunneling*: transportar datagramas IPv6 como *payload* de datagramas IPv4
  - Permite desplegar gradualmente IPv6 y continuar operando con routers IPv4
  - Esta técnica se usa en otros contextos (e.g. redes celulares 4G/5G)

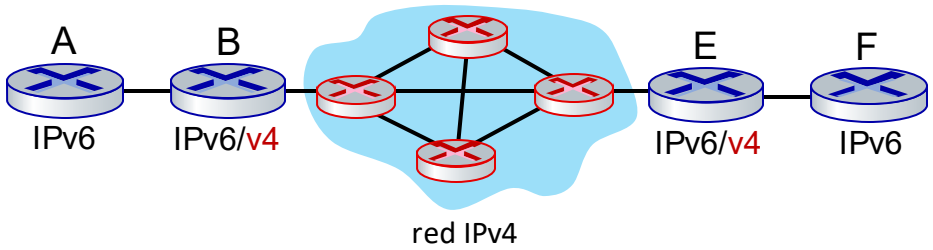


Tunneling y encapsulamiento

Ethernet conectando dos routers IPv6

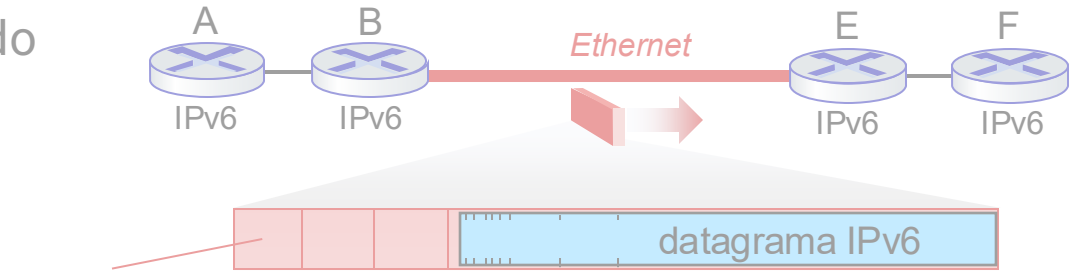


Red IPv4 conectando dos routers IPv6



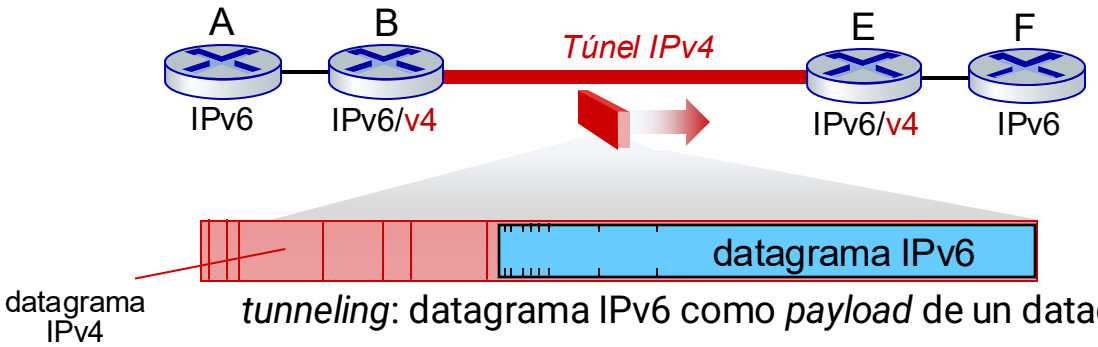
Tunneling y encapsulamiento

Ethernet conectando dos routers IPv6



lo típico: datagrama como *payload* del *frame* de nivel de enlace

Túnel IPv4 conectando dos routers IPv6

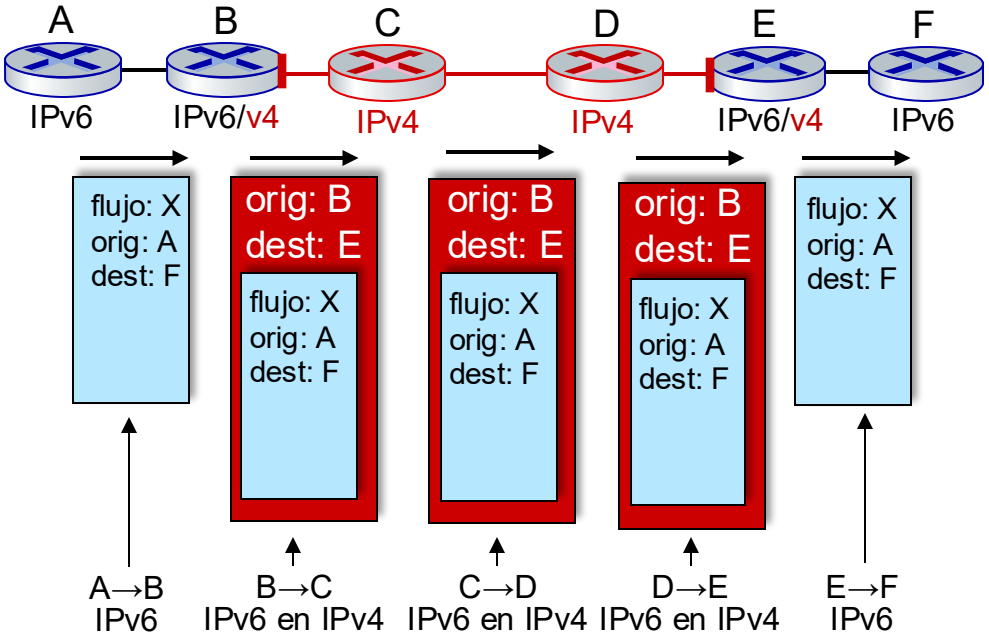


tunneling: datagrama IPv6 como *payload* de un datagrama IPv4

Tunneling

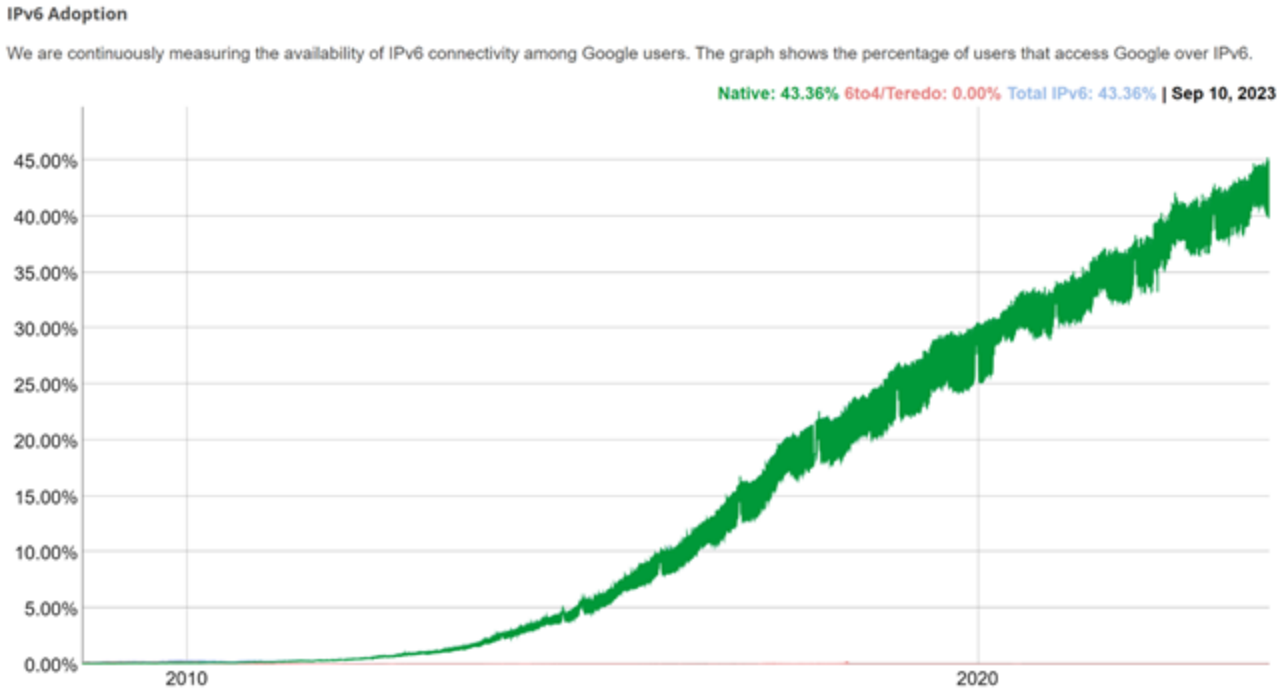
vista lógica

vista física



Adopción de IPv6

- Google: **~44%** de los clientes acceden a los servicios vía IPv6 a octubre 2025 (en Argentina, **~22%**)



Adopción de IPv6

- El tiempo de despliegue es muy largo (**+25** años y contando)
- Es **muy complicado** cambiar protocolos de nivel de red: los “cimientos” de Internet
- Es más simple implementar cambios en los protocolos de nivel de aplicación (e.g. WWW, redes sociales, *streaming* –en permanente evolución y desarrollo)

Software Defined Networking

Forwarding generalizado: *match-plus-action*

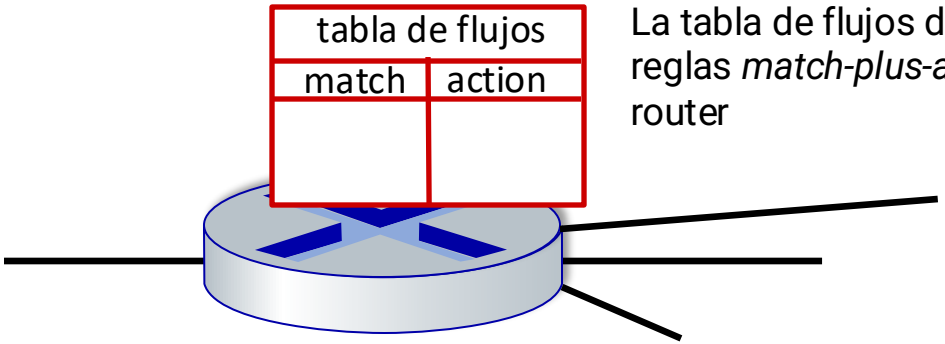
Los routers tienen una tabla de forwarding (aka: tabla de flujos)

- Abstracción "**match-plus-action**": corresponder bits en los paquetes entrantes y luego tomar acciones

  - *Destination-based forwarding:* reenviar paquetes a partir de su IP destino
  - *Forwarding generalizado*:
    - La acción puede determinarse a partir de varios campos del header
    - Múltiples acciones posibles: descartar, copiar, modificar, logging, etc.

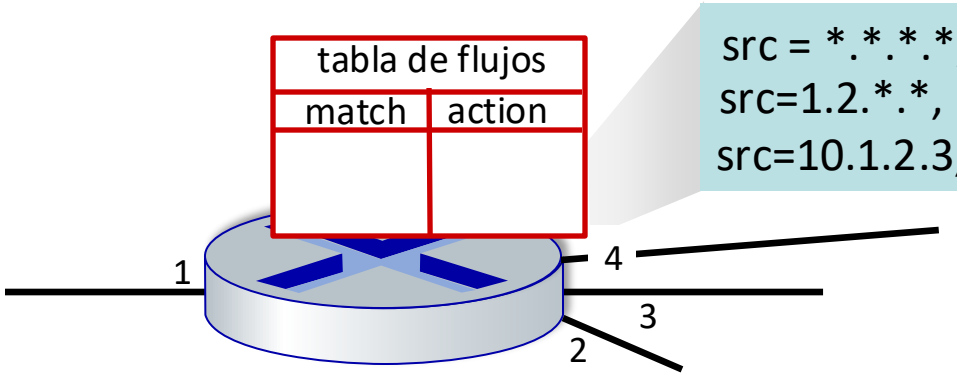
Abstracción: tabla de flujos

- Flujo: definido por los campos de los *headers* (de los niveles de enlace, red y/o transporte)
- *Forwarding* generalizado: reglas de manejo de paquetes
  - *match*: patrones de valores en los campos de los *headers*
  - *actions*: sobre los paquetes coincidentes –descartar (*drop*), reenviar, modificar, enviar al controlador
  - prioridad: para desambiguar entre patrones solapados
  - contadores: #bytes y #paquetes



La tabla de flujos define las reglas *match-plus-action* del router

Abstracción: tabla de flujos

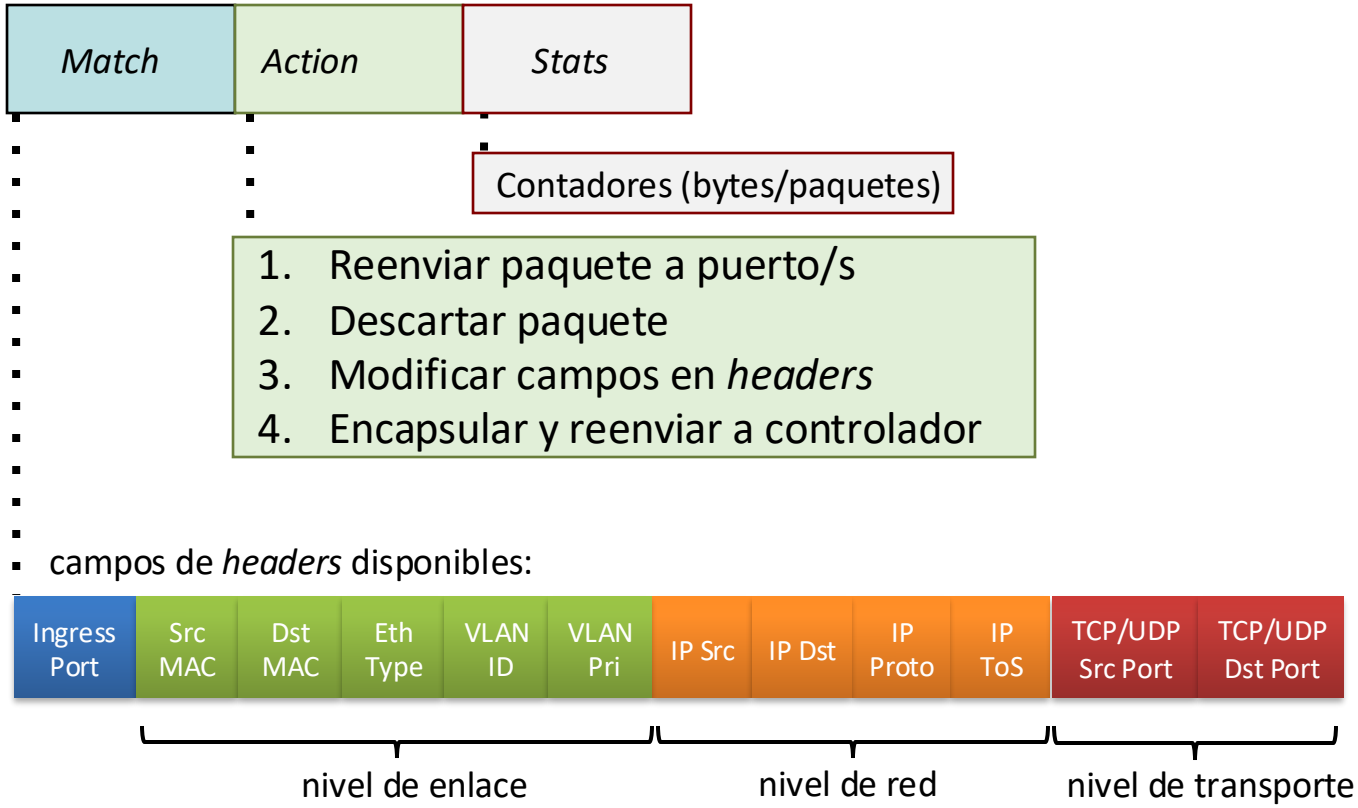


src = *.*.*.*, dest=3.4.*.*	reenviar(2)
src=1.2.*.*, dest=*.*.*.*	descartar
src=10.1.2.3, dest=*.*.*.*	a controlador

El protocolo OpenFlow

- **OpenFlow** es un protocolo abierto que permite la separación del plano de control y el plano de datos
- Pionero en los conceptos de *forwarding generalizado* y SDN
- Primera versión liberada en febrero de 2011
- **Controlador**: el *cerebro* de una red SDN; se comunica con los routers para tomar decisiones de ruteo y gestión del tráfico
  - Instala y configura tablas de flujos en los routers
- Corre sobre TCP (el controlador escucha en el puerto 6653)

OpenFlow: entradas de la tabla de flujos



OpenFlow: ejemplos

Destination-based forwarding

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	**port6**

Los datagramas IP destinados a la dirección 51.6.0.8 deben ser reenviados al puerto de salida 6

Firewall

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	**drop**

Descartar todos los datagramas destinados al puerto TCP 22 (ssh)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	**drop**

Descartar todos los datagramas enviados por el host 128.119.1.1

Abstracción de OpenFlow

- La abstracción *match+action* permite unificar distintos tipos de dispositivos de red

Router
- *match:* prefijo de la dirección IP destino más largo
- *action:* reenviar hacia un enlace de salida

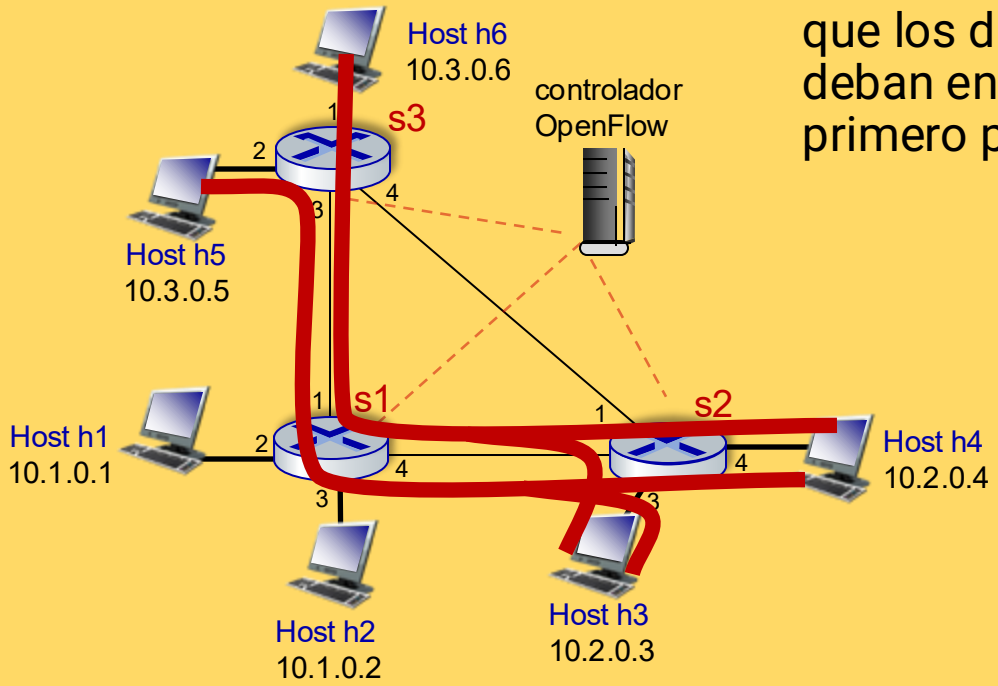
Switch
- *match:* dirección destino de nivel de enlace
- *action:* reenviar o *floodear*

Firewall
- *match*: direcciones IP y puertos TCP/UDP
- *action:* permitir o denegar

NAT
- *match:* dirección IP y puerto
- *action:* reescribir dirección y puerto

Ejercicio!

- Proponer tablas de flujos para los routers de la figura de forma tal que los datagramas de h_5 y h_6 que deban enviarse a h_3 o h_4 pasen primero por s_1 y luego por s_2



Ejercicio!

- Proponer tablas de flujos para los routers de la figura de forma tal que los datagramas de h_5 y h_6 que deban enviarse a h_3 o h_4 pasen primero por s_1 y luego por s_2