

Routing: NATs

Area de Ingeniería Telemática
<http://www.tlm.unavarra.es>

Grado en Ingeniería en Tecnologías de
Telecomunicación, 3º

Temas de teoría

1. Introducción
2. QoS
3. Encaminamiento dinámico en redes IP
4. Tecnologías móviles
5. Otros temas

Objetivos

- Comprender el funcionamiento general de los NAT
- Conocer los problemas que pueden dar a las aplicaciones

Contenido

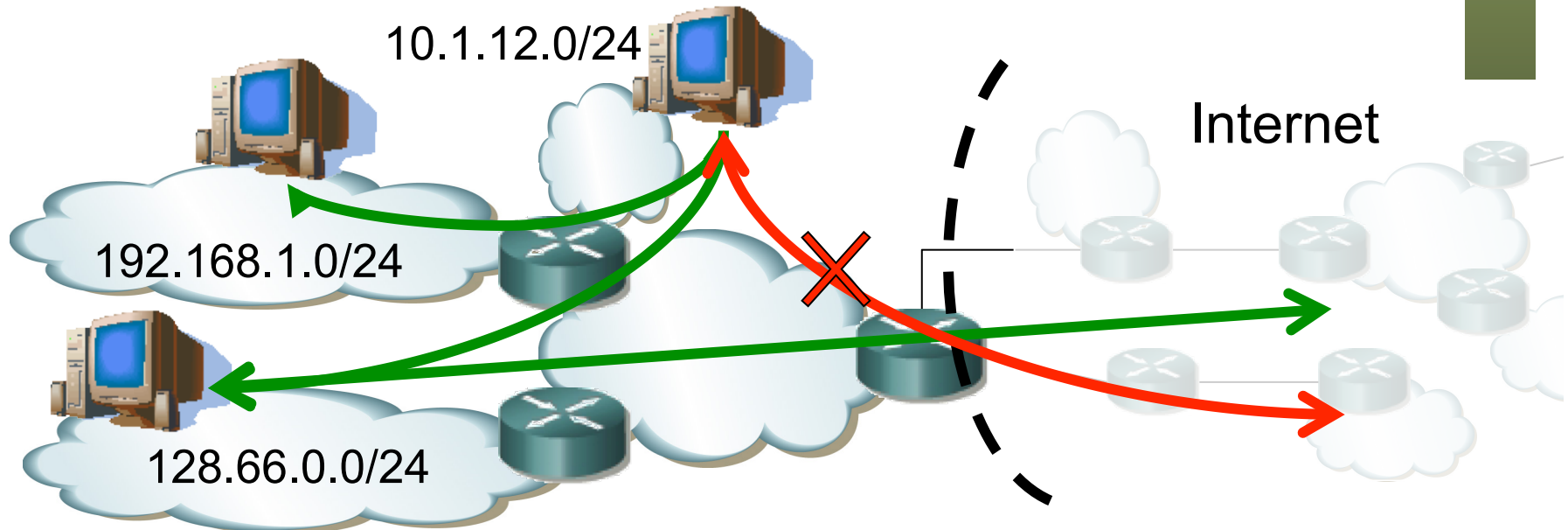
- Introducción
- Funcionamiento básico
- NATs y aplicaciones
- Recomendaciones
- Otros aspectos

Problemas de IPv4

- Escasez de direcciones
 - Desaprovechamiento con Classful:
 - Clase A: Más de 16M de direcciones
 - Clase B: 64K direcciones
 - Con CIDR:
 - Hemos llegado de nuevo al problema de agotamiento
 - También PCs que se usen esporádicamente
- Complejidad innecesaria en los routers
- Algunas soluciones:
 - DHCP (escasa solución)
 - IPv6 (lejana solución)
 - NAT (vino a nuestro rescate... según cómo se mire)

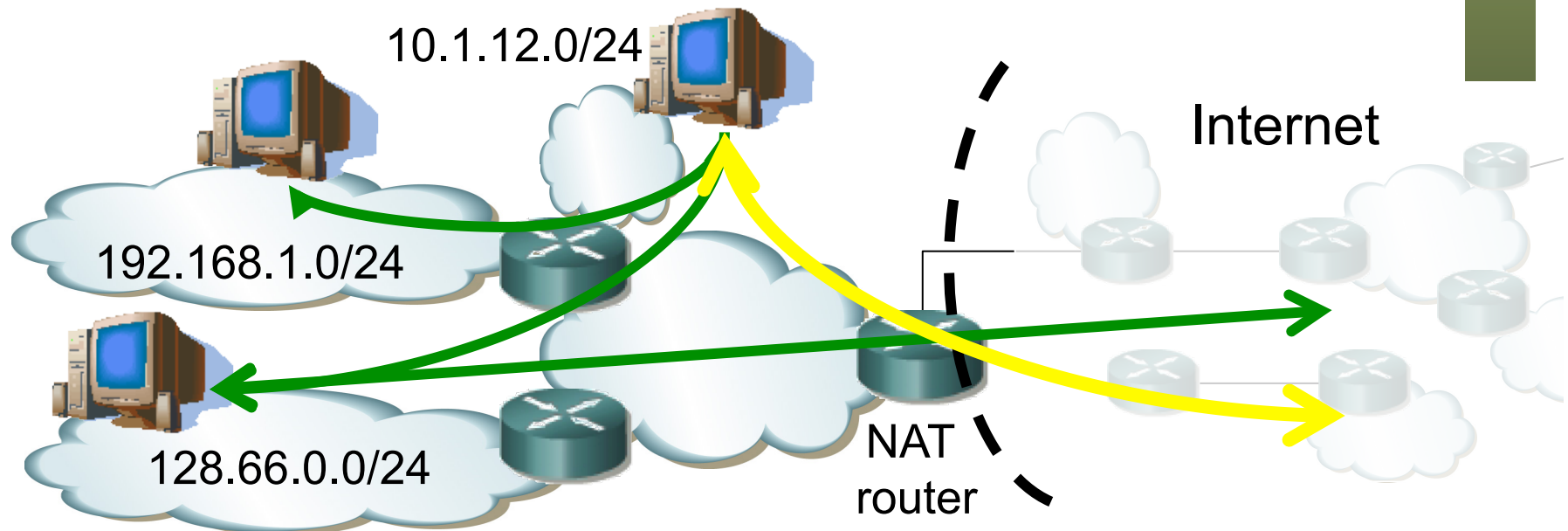
Direccionamiento privado

- 10.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12
- Pueden comunicarse con cualquier máquina de la red interna
- Al exterior solo pueden salir paquetes IP con direcciones públicas únicas



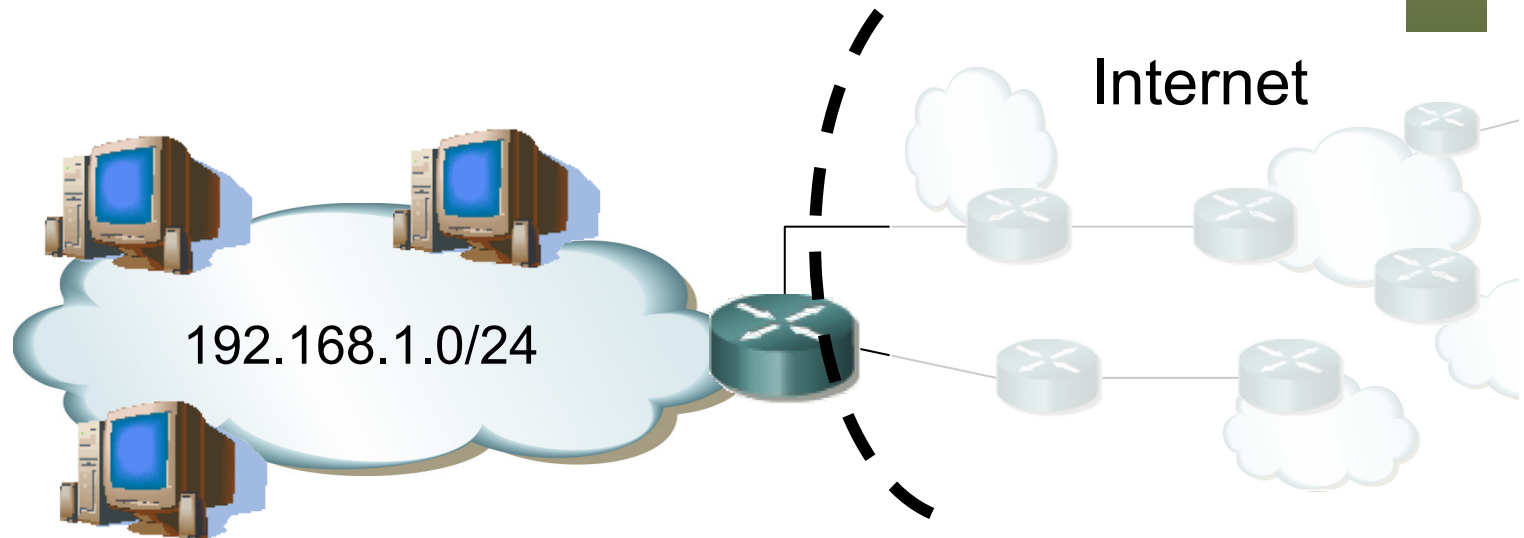
NATs

- Habilitan esa comunicación
- En los paquetes IP el NAT cambiará la dirección privada por una pública
- Escenario más conocido (...)



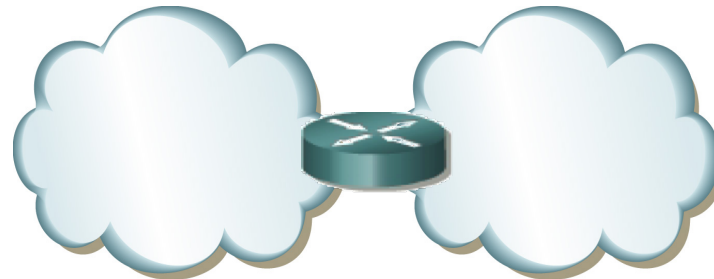
NATs

- Habilitan esa comunicación
- En los paquetes IP el NAT cambiará la dirección privada por una pública
- Escenario más conocido: Usuario residencial



Introducción

- Hoy en día hay ya varias RFCs tratando el tema de los NATs
- Por ejemplo:
 - RFC 3022 “Traditional IP Network Address Translator (Traditional NAT)
 - RFC 2663 “IP Network Address Translator (NAT) Terminology and Considerations”
 - Y otras que comentaremos más adelante
- Un NAT mapea direcciones entre dos dominios
- Se habla de NATs y NAPT (a veces PATs) aunque por extensión se les suele llamar a ambos NATs
- Se dice que hacen *transparent routing*, enrutando paquetes entre dos dominios



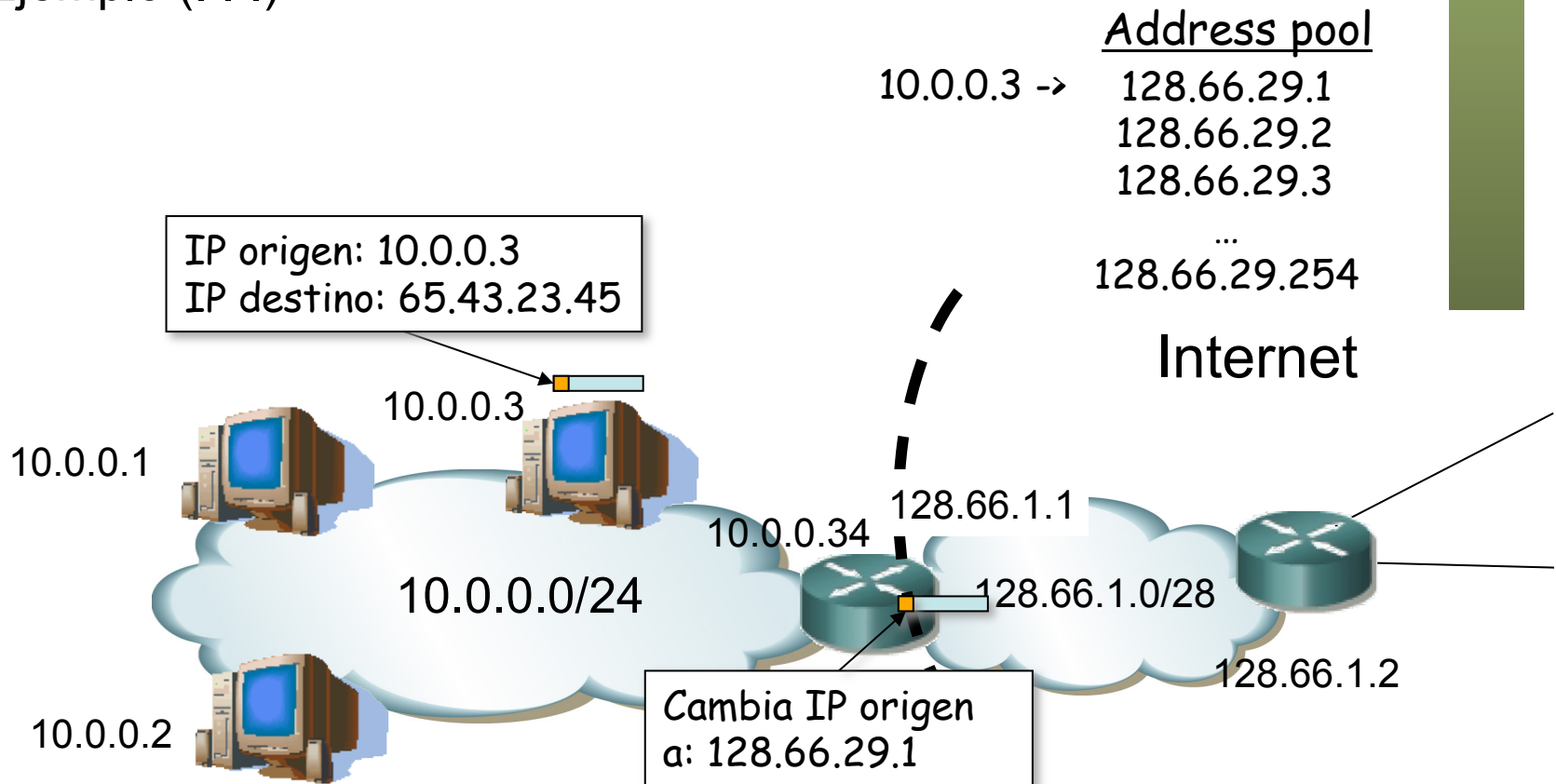
End-to-end principle

- Siempre que sea posible implementar los protocolos en los extremos de la red
- Implementar en la red lo menos posible
- Red con mínima inteligencia (la red es difícil de cambiar)
- Inteligencia en los extremos (es más sencillo añadir nueva funcionalidad)
- La Internet es un ejemplo, con IP en la red y el resto de protocolos solo en los extremos
- NATs rompen el funcionamiento extremo-a-extremo de la Internet
- Eso va a dar problemas a las aplicaciones, a su funcionamiento y a su despliegue
- Veremos varios escenarios con ejemplos



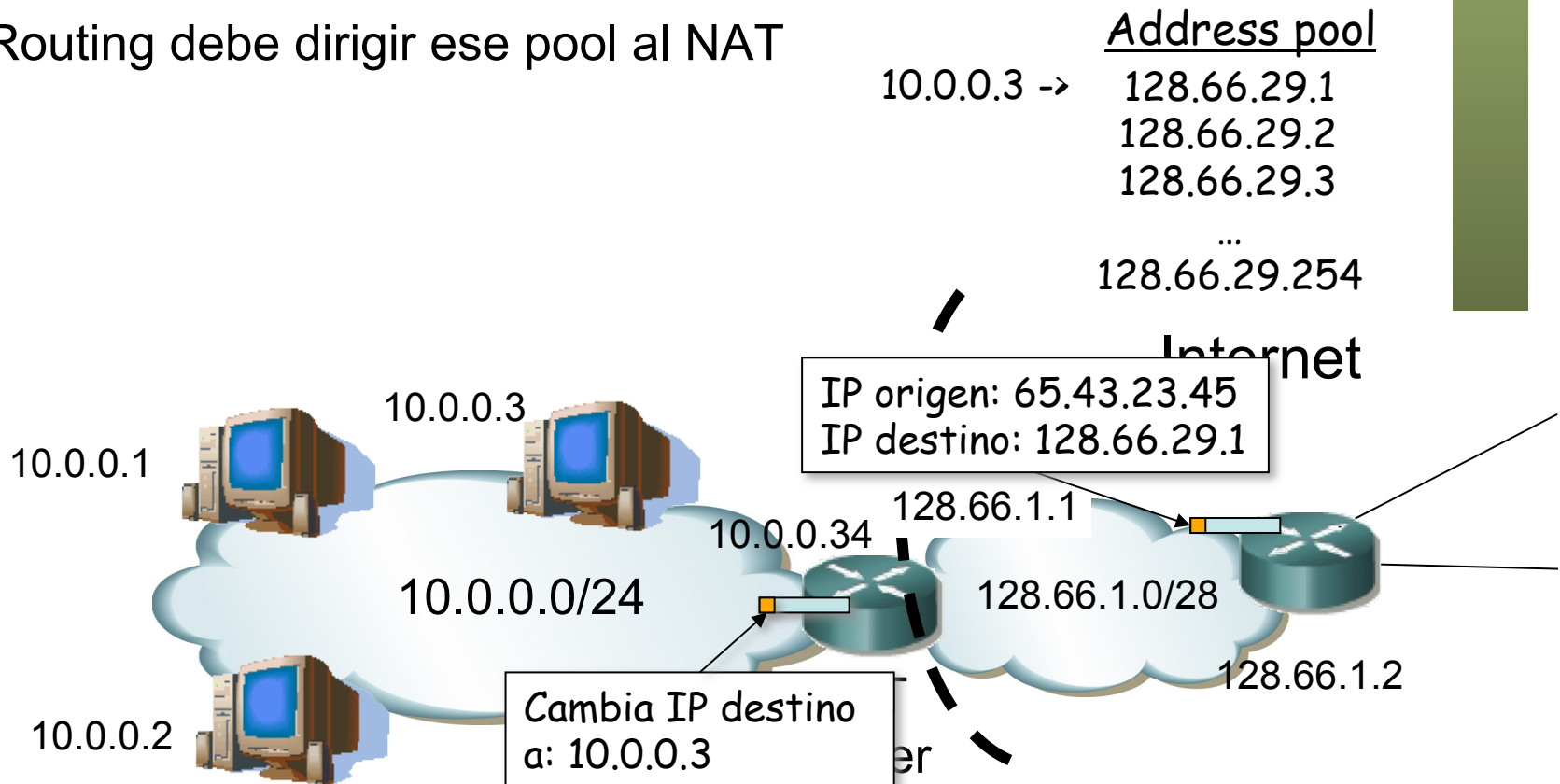
NATs

- NAT tiene asignado un bloque (*pool*) de direcciones públicas
- Cuando reenvía al exterior un paquete cambia la dirección origen por una del pool
- Apunta el *mapeo* para aplicarlo en sentido contrario
- Ejemplo (. . .)



NATs

- Cuando venga un paquete de esa dirección IP externa vendrá dirigido a la dirección que colocó como origen el router NAT
- La tabla de mapeos indica el cambio a hacer (... ..)
- Para el host remoto el flujo es con la dirección pública pues nunca ve la privada
- Routing debe dirigir ese pool al NAT



NATs: mapeo

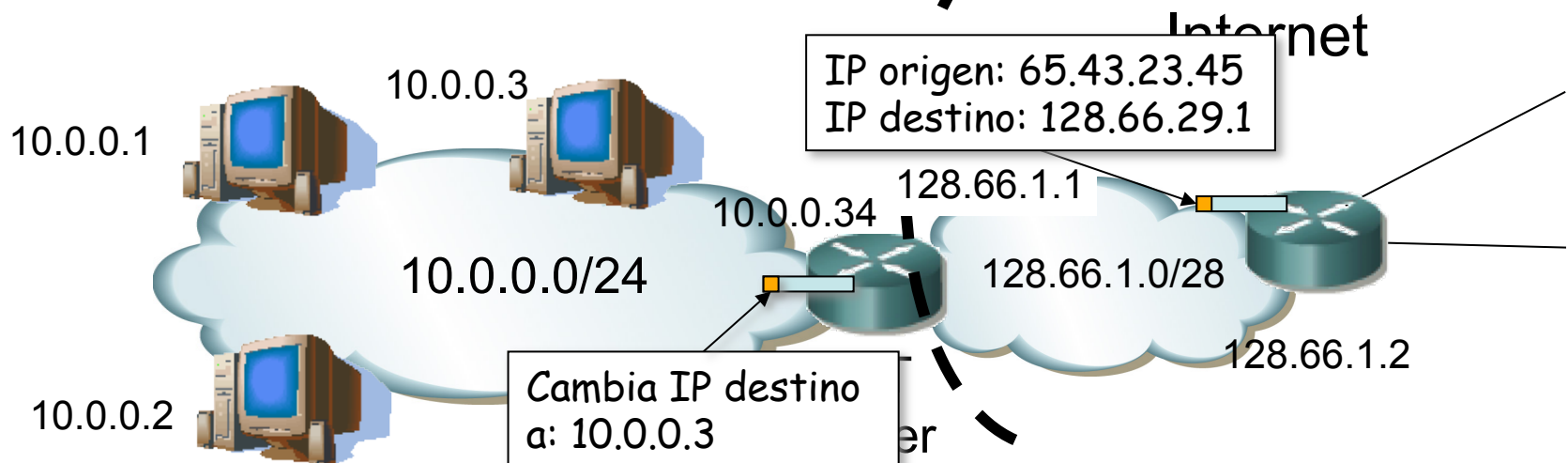
- Estático
 - Preconfigurado 1 a 1
 - Requiere tantas direcciones como hosts con direccionamiento privado
- Dinámico
 - Mapea bajo demanda
 - Requiere menos direcciones públicas
 - Un timer de inactividad para eliminar el mapeo

Address pool

128.66.29.1
 128.66.29.2
 128.66.29.3

...

128.66.29.254



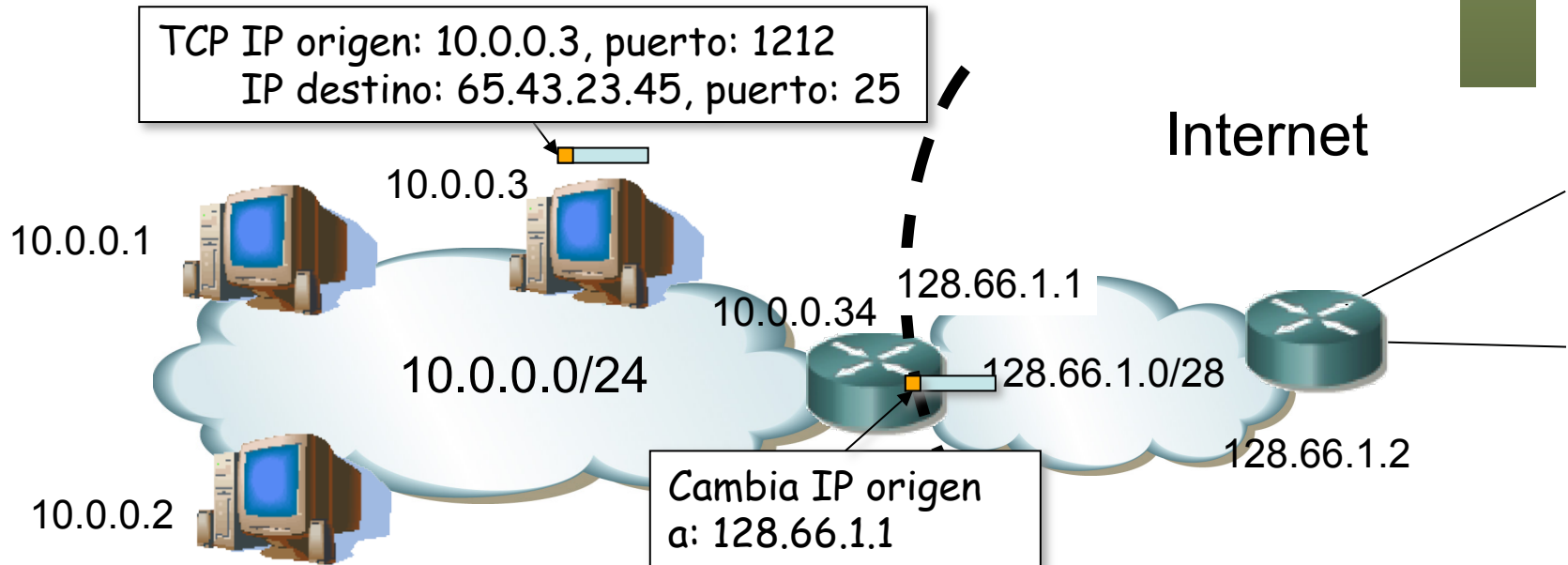
NAPT

- Network Address/Port Translator
- Va a poder modificar también la cabecera del protocolo de transporte
- Solo para TCP, UDP e ICMP
- “Sesiones”
 - TCP/UDP (TU): {(IP-1, Port-1), (IP-2, Port-2)}
 - ICMP: (IP-1, queryID, IP-2)
 - El concepto de sesión a nivel de aplicación puede diferir e incluir varias de éstas
 - En TCP termina tras intercambio de FINs/RST aunque pueden perderse y se mantiene durante un tiempo (recomendado 4min)
 - Hosts pueden reiniciarse así que siempre deben caducar los mapeos tras un tiempo de inactividad

NAPT

- Pocas direcciones públicas, por ejemplo solo una (que puede ser la de su interfaz exterior)
- Paquete hacia el exterior provoca nuevo mapeo (. . .)

Proto.	Interno	Pública	Exterior
TCP	10.0.0.3:1212	128.66.1.1:1212	65.43.23.45:25

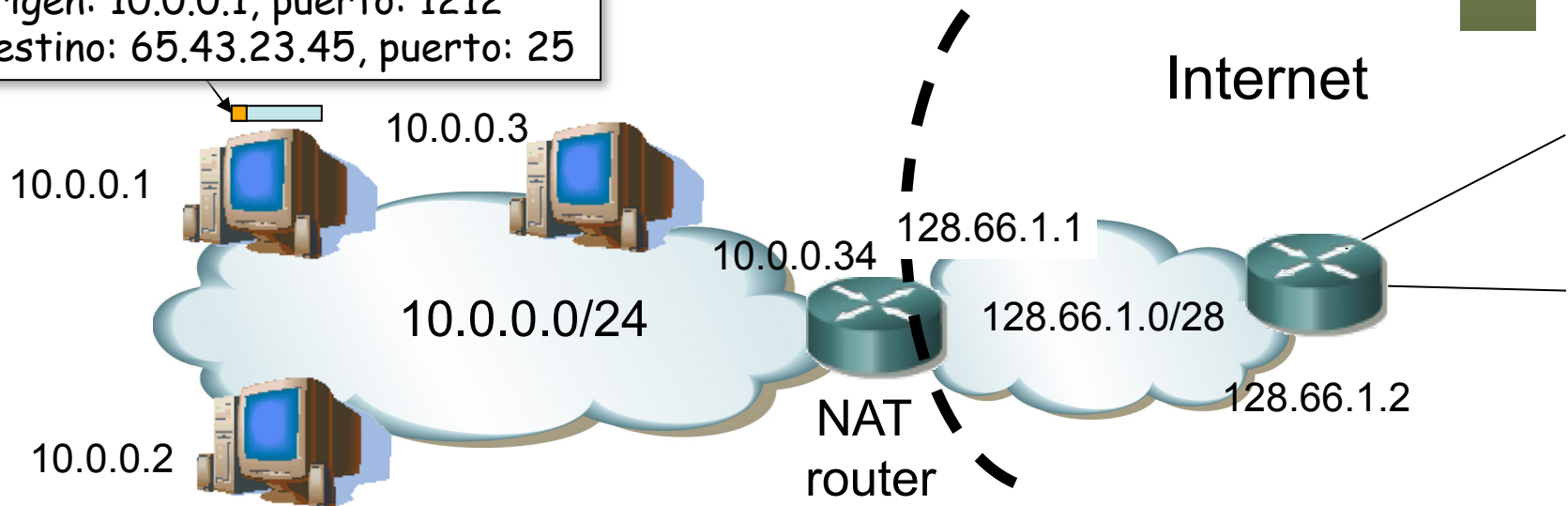


NAPT

- Otro host podría ir al mismo servidor y servicio empleando el mismo puerto local (no hay coordinación entre ellos)
- El mapeo provoca una colisión (. . .)

Proto.	Interno	Pública	Exterior
TCP	10.0.0.3:1212	128.66.1.1:1212	65.43.23.45:25

TCP IP origen: 10.0.0.1, puerto: 1212
 IP destino: 65.43.23.45, puerto: 25

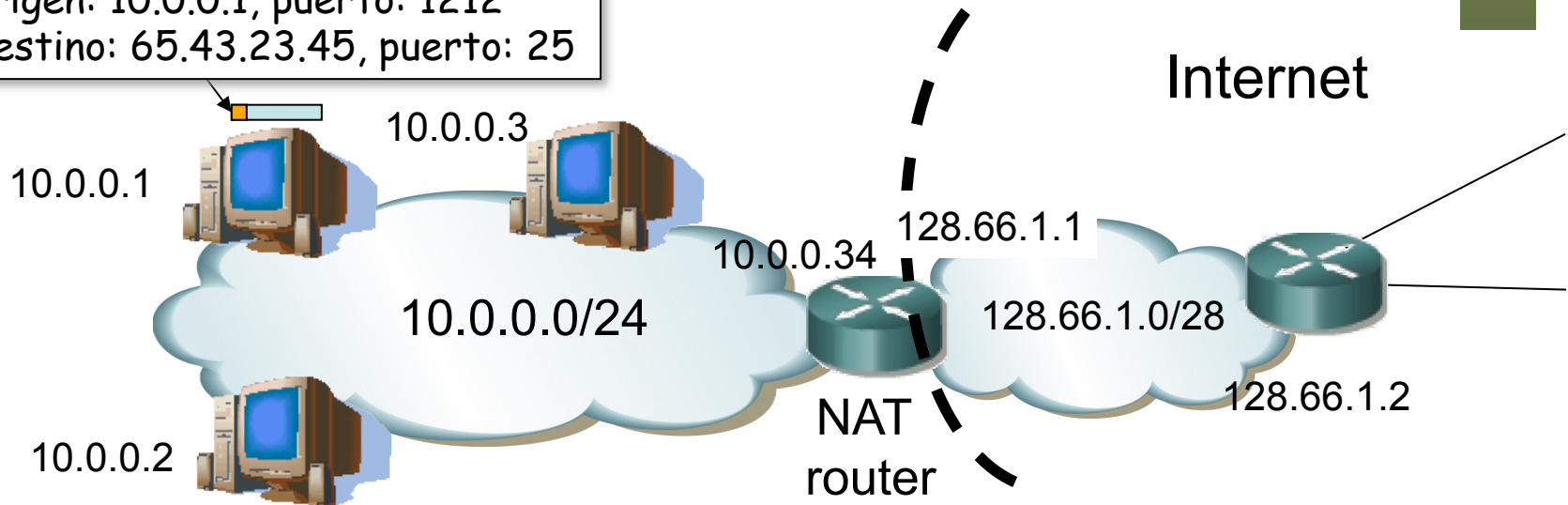


NAPT

- Otro host podría ir al mismo servidor y servicio empleando el mismo puerto local (no hay coordinación entre ellos)
- El mapeo provoca una colisión (. . .)

Proto.	Interno	Pública	Exterior
TCP	10.0.0.3:1212	128.66.1.1:1212	65.43.23.45:25
TCP	10.0.0.1:1212	128.66.1.1:1212	65.43.23.45:25

TCP IP origen: 10.0.0.1, puerto: 1212
 IP destino: 65.43.23.45, puerto: 25

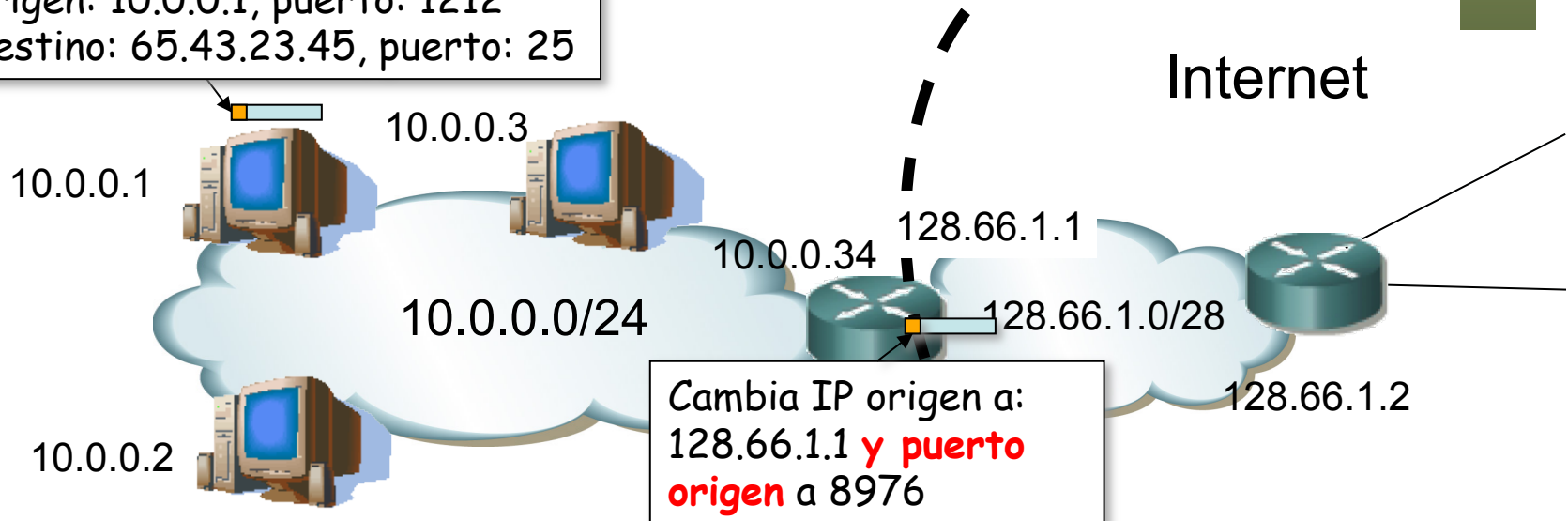


NAPT

- El NAPT va a cambiar también el puerto origen

Proto.	Interno	Pública	Exterior
TCP	10.0.0.3:1212	128.66.1.1:1212	65.43.23.45:25
TCP	10.0.0.1:1212	128.66.1.1:1212	65.43.23.45:25
TCP	10.0.0.1:1212	128.66.1.1: 8976	65.43.34.45:25

TCP IP origen: 10.0.0.1, puerto: 1212
 IP destino: 65.43.23.45, puerto: 25

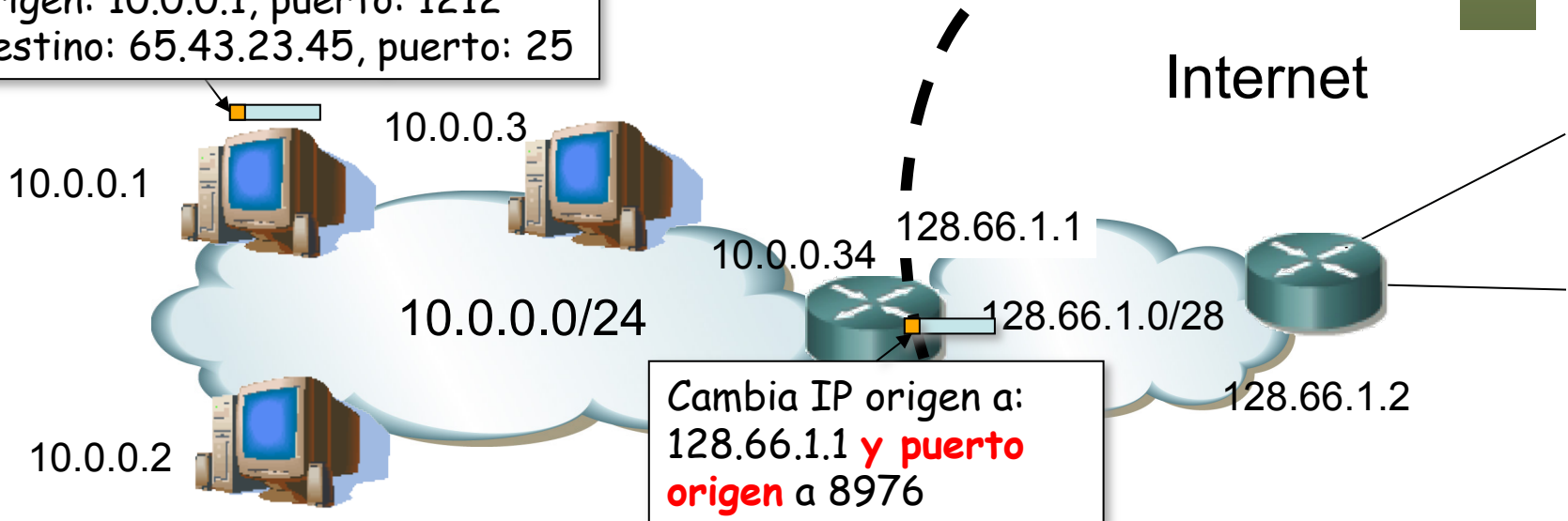


NAPT: problema

- Ante una dirección IP y puerto externo muy popular hay un límite de mapeos
- Se debe al límite de puertos TCP disponibles (16 bits)

Proto.	Interno	Pública	Exterior
TCP	10.0.0.3:1212	128.66.1.1:1212	65.43.23.45:25
TCP	10.0.0.1:1212	128.66.1.1:1212	65.43.23.45:25
TCP	10.0.0.1:1212	128.66.1.1: 8976	65.43.34.45:25

TCP IP origen: 10.0.0.1, puerto: 1212
 IP destino: 65.43.23.45, puerto: 25

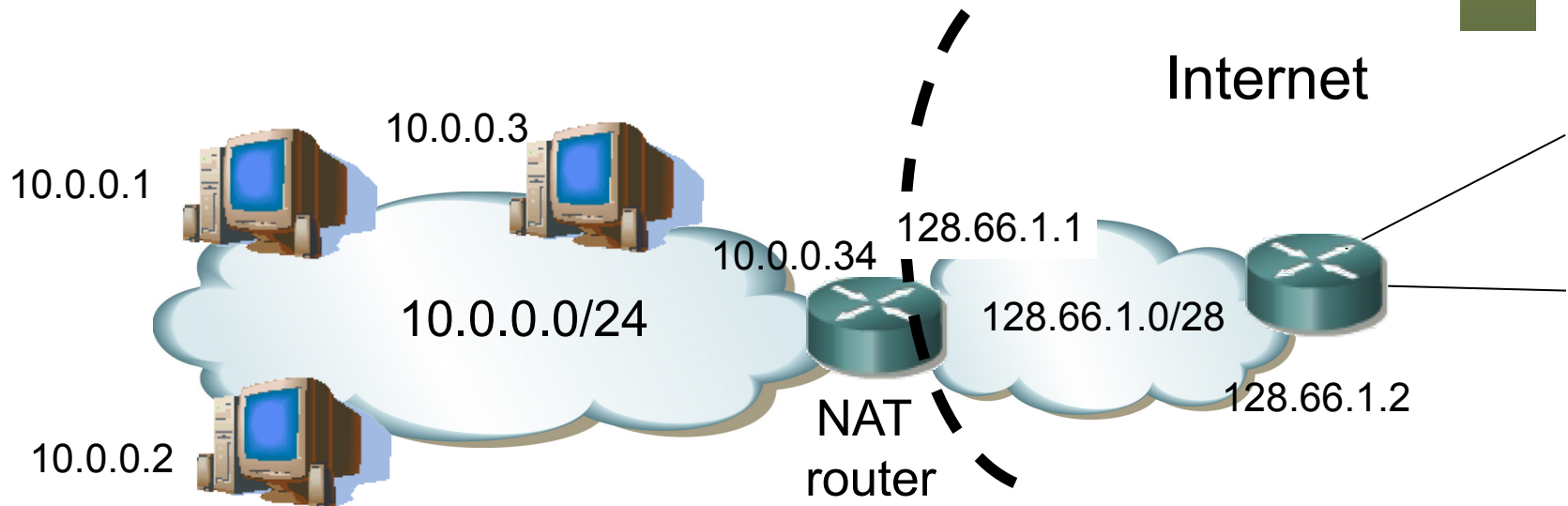


Cambia IP origen a:
 128.66.1.1 **y puerto**
origen a 8976

Conexiones entrantes

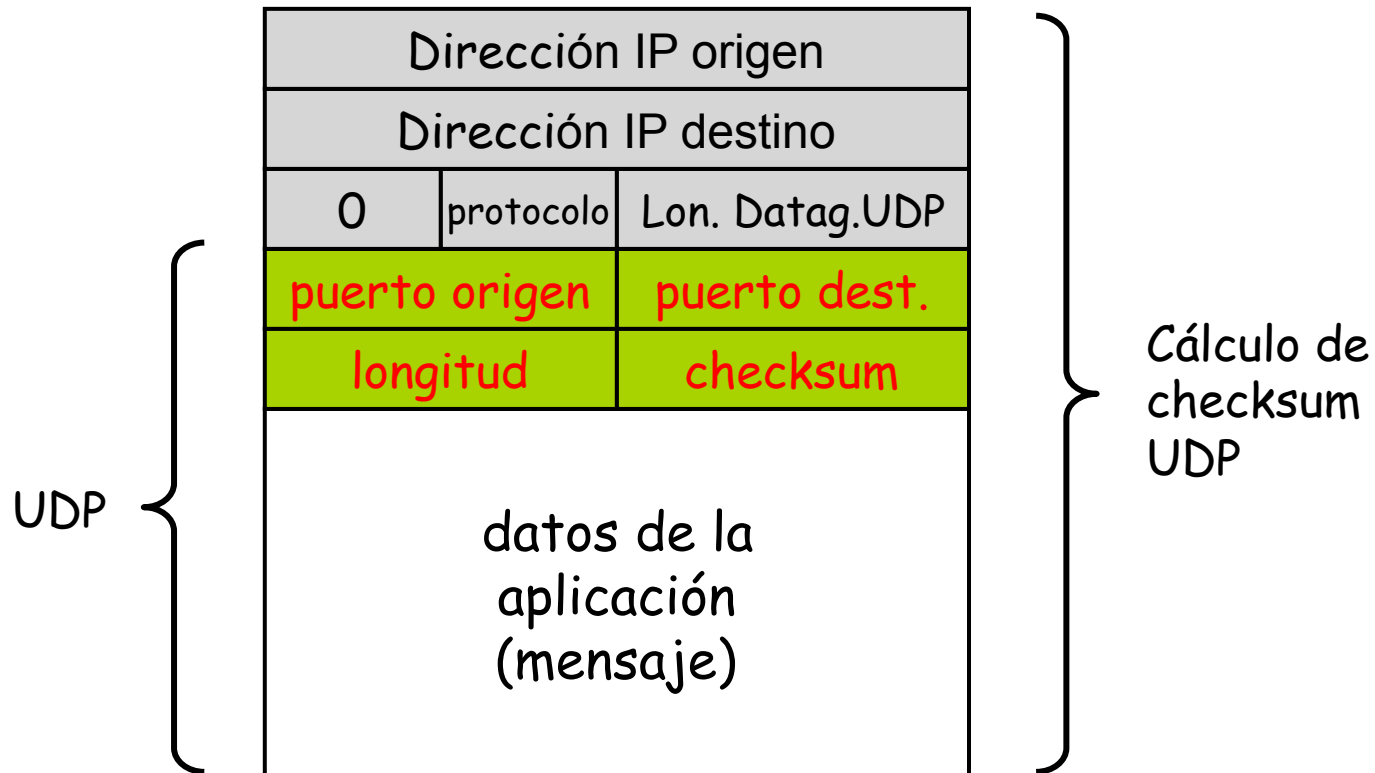
- Normalmente mediante mapeo estático
- Cambiando o no el puerto local

Proto.	Interno	Pública	Exterior
TCP	10.0.0.3:1212	128.66.1.1:80	65.43.23.45:80



Checksums

- Cambio en dirección IP requiere recalcular checksum IP
- También requiere recalcular checksum TCP/UDP pues protege también a las direcciones IP



NATs y aplicaciones

- Problemas con logs y estadísticas en servidores
 - El mismo host puede aparecer en el exterior con diferente dirección en diferente momento
 - Varios hosts pueden aparecer con la misma dirección
 - Con datos de red+transporte no se pueden distinguir
 - Dificulta hacer estadísticas por usuario o identificar responsables de abusos
- Problemas con aplicaciones que
 - En la comunicación de datos hablan de direcciones IP y/o puertos (FTP, SIP, H.323, juegos online...)
 - Esa información es necesaria para establecer otras sesiones
 - Se emplean entonces ALGs (...)

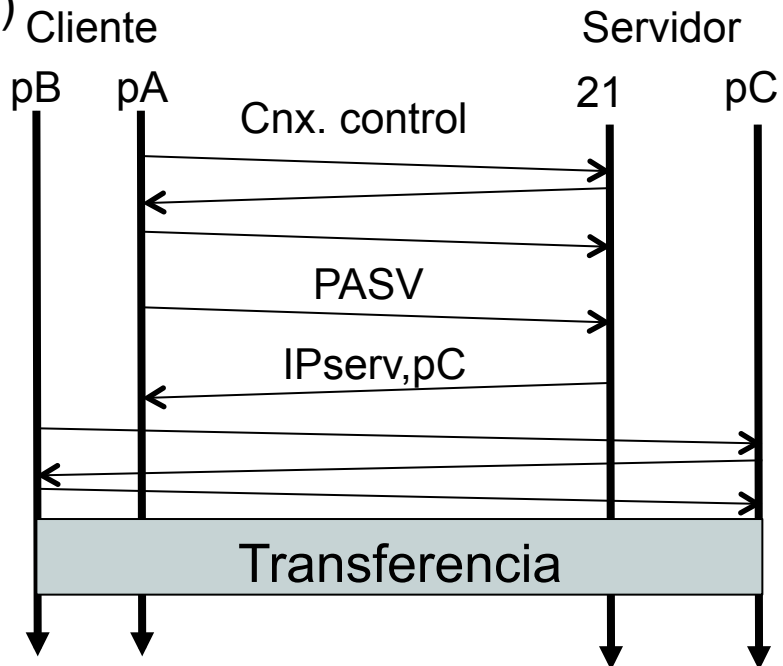


ALGs

- *Application Level Gateways*
- Parte del NAT/NAPT
- Monitoriza y modifica el payload (datos TCP/UDP)
- Deben conocer el protocolo de nivel de aplicación
- No debe estar encriptado (o el ALG debe tener la clave)
- Ejemplo: FTP

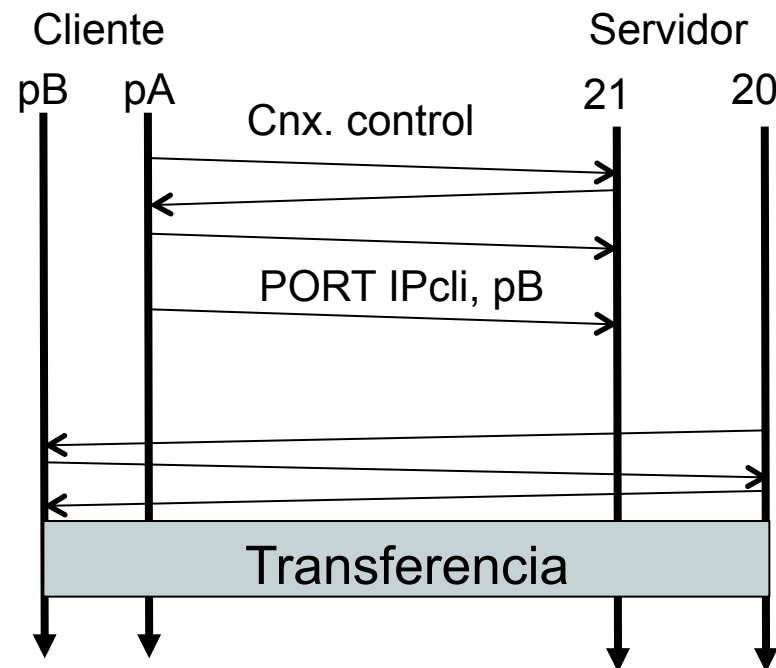
FTP: File Transfer Protocol

- RFC 959
- Servidor emplea puerto TCP 21
- Cliente establece una conexión de control con el servidor
- Transferencia en modo **pasivo**
 - Cliente envía comando a servidor (...)
 - Servidor contesta indicando la dirección IP y puerto en que espera conexión (...)
 - Cliente establece una conexión con el servidor a ese puerto (...)
 - Se produce la transferencia (...)
 - El servidor tiene que aceptar conexiones en múltiples puertos
 - Podría ser un problema con firewalls



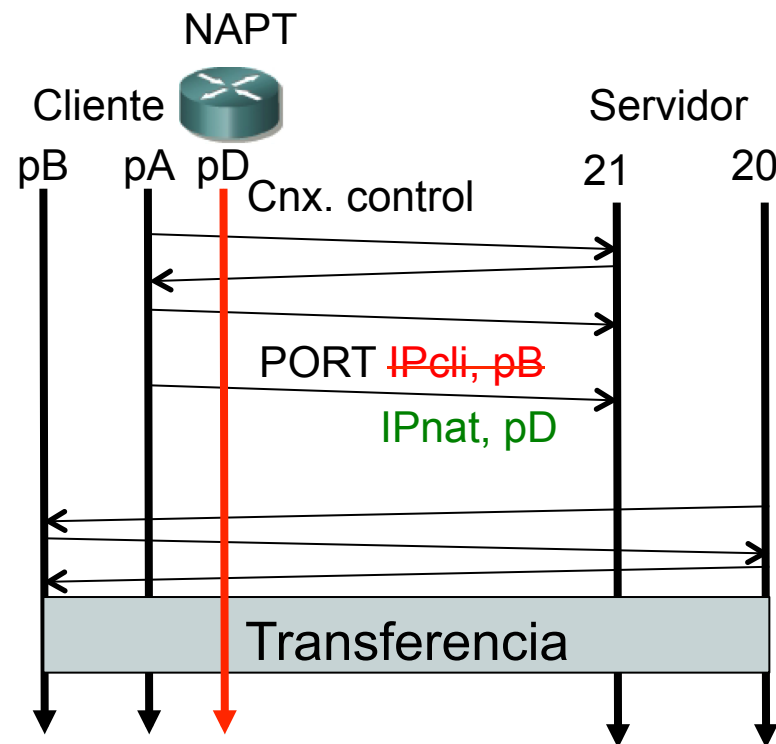
FTP: File Transfer Protocol

- RFC 959
- Servidor emplea puerto TCP 21
- Cliente establece una conexión de control con el servidor
- Transferencia en modo **activo**
 - Cliente envía comando a servidor indicando dirección IP y puerto en que espera conexión (...)
 - Servidor establece una conexión con el cliente a ese puerto (...)
 - Se produce la transferencia (...)



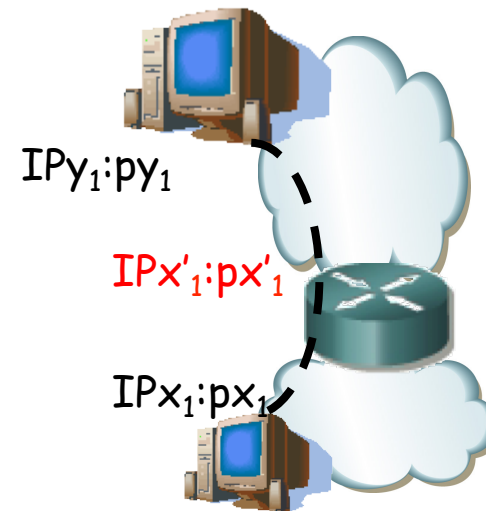
NAT y FTP activo

- El cliente ha especificado un puerto local, así como su dirección
- NAT debe seguir el stream de datos para reconocer el comando
- Reconstruir el stream si el comando está fragmentado
- Modificarlo con dirección externa y puerto que seleccione
- Introducir mapeo para esa (dirección,puerto)
- La modificación puede introducir más o menos bytes en el comando FTP (son ASCII)
- Entonces debe modificar los números de secuencia TCP y de ACK a partir de ahí



UNSAF

- UNilateral Self-Address Fixing (UNSAF)
- En ocasiones la aplicación necesita saber con qué dirección IP y puerto externo está comunicándose con el otro extremo
- Son métodos para que un extremo averigüe o fije la dirección y puerto con la que le puede acceder otro extremo
- Es decir, saber los valores de dirección y puerto públicos que va a emplear el NAT en la comunicación, tal vez para comunicarla en los datos del protocolo
- Se basan en heurísticos

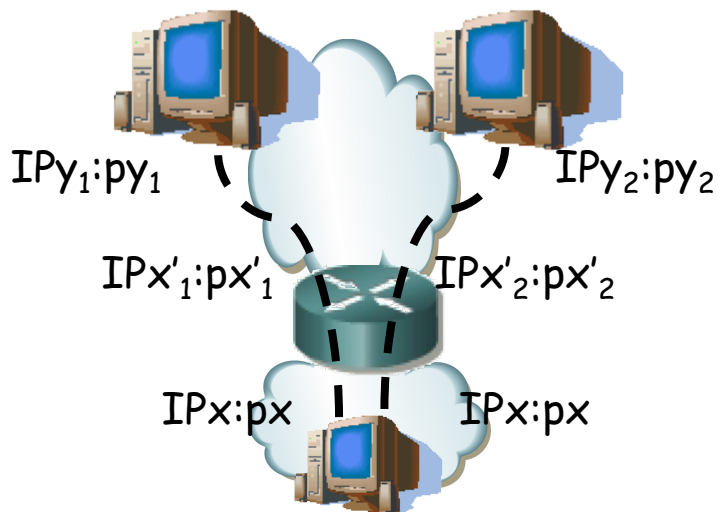


REQs UDP: RFC 4787

- A.k.a. BCP 127 (Enero 2007)
- “Network Address Translation (NAT) Behavioral Requirements for Unicast UDP”
- Contiene Requerimientos y Recomendaciones
- No quiere decir eso que todas las implementaciones las sigan (¡ni mucho menos!)
- Veamos algunas de ellas (...)

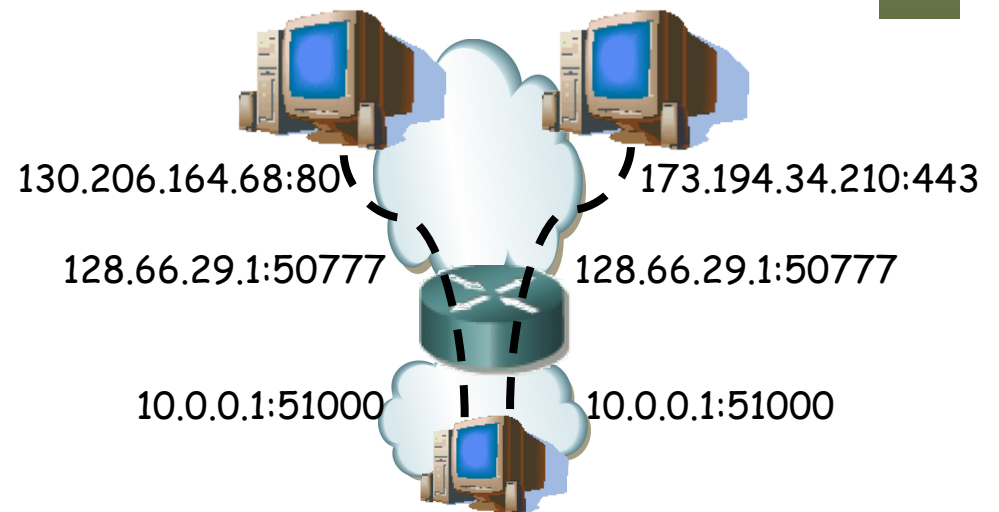
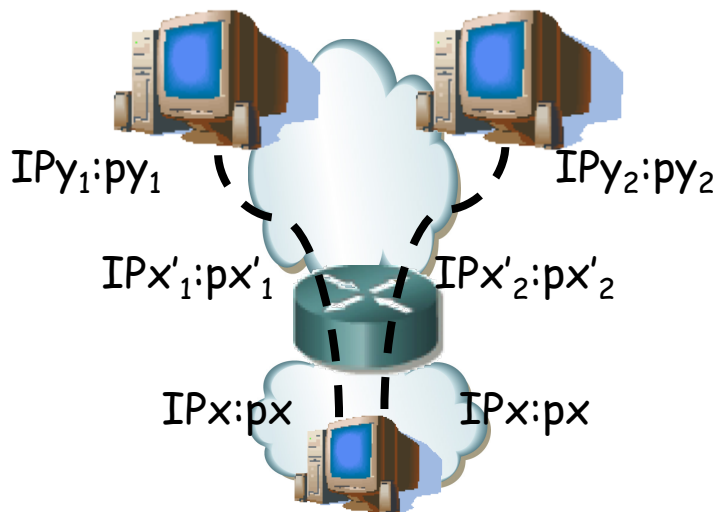
Mapeo de dirección y puerto

- Mapeo establecido: $(IPx:px, IPx'_1:px'_1) \rightarrow IPy_1:py_1$
- A continuación se quiere enviar de $IPx:px$ a $IPy_2:py_2$
- Es importante cómo es el comportamiento del NAT cuando hay múltiples sesiones con diferentes terminaciones externas



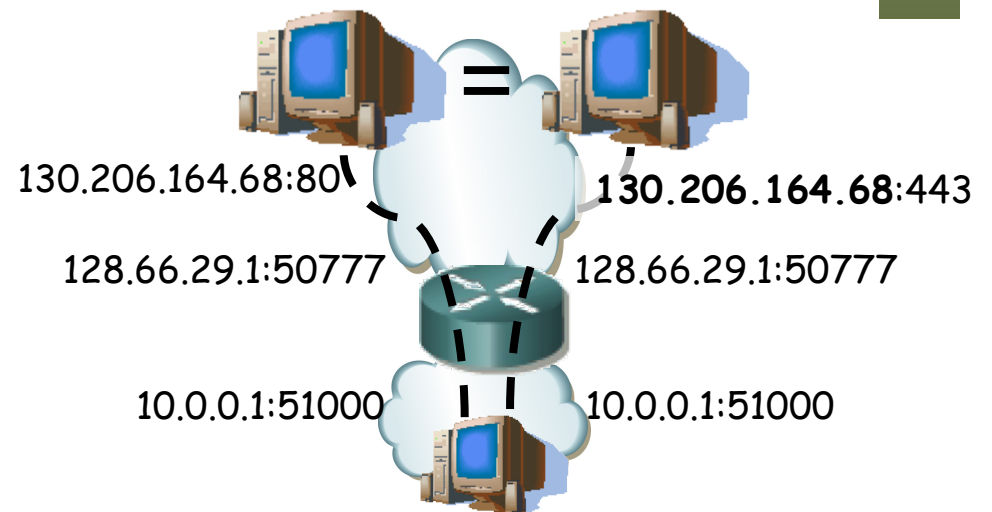
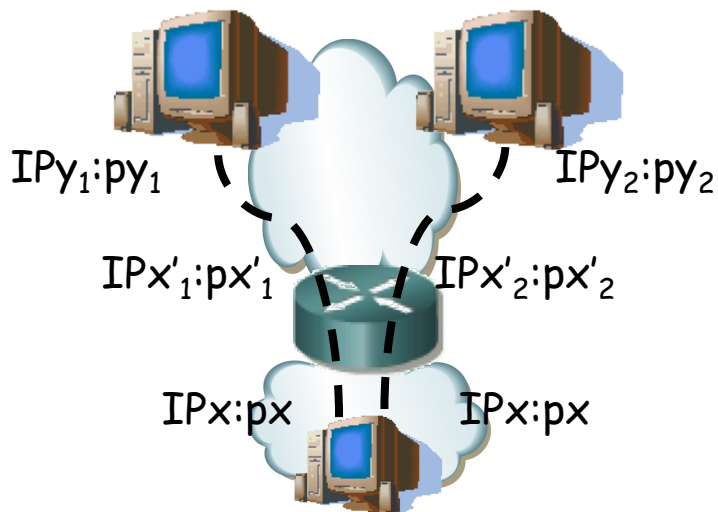
Mapeo de dirección y puerto

- Mapeo establecido: $(IPx:px, IPx'_1:px'_1) \rightarrow IPy_1:py_1$
- A continuación se quiere enviar de $IPx:px$ a $IPy_2:py_2$
- *Endpoint-Independent Mapping*
 - Reutiliza el mapeo establecido con esa dirección y puerto internos a cualquier dirección IP y puerto externos
 - Es decir, $IPx'_1:px'_1 = IPx'_2:px'_2$ para cualquier valor de $IPy'_2:py'_2$



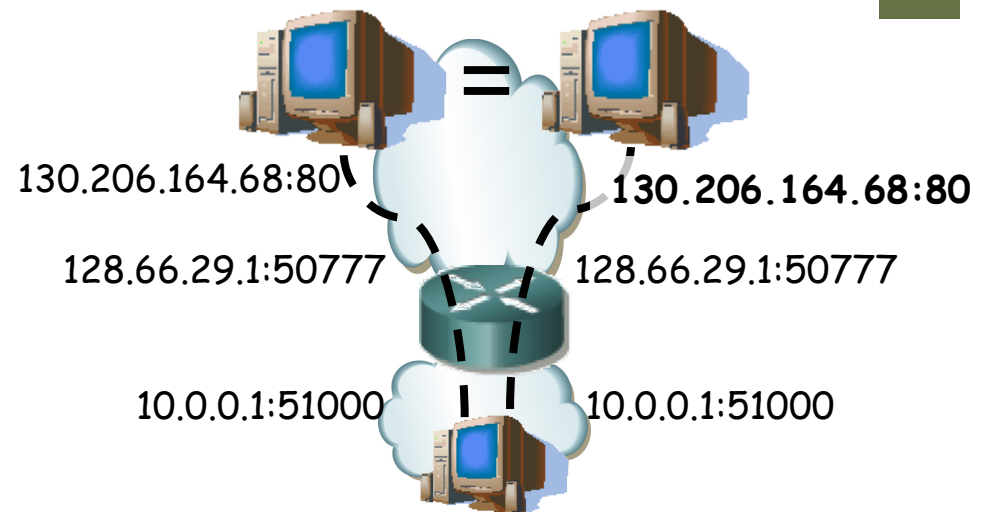
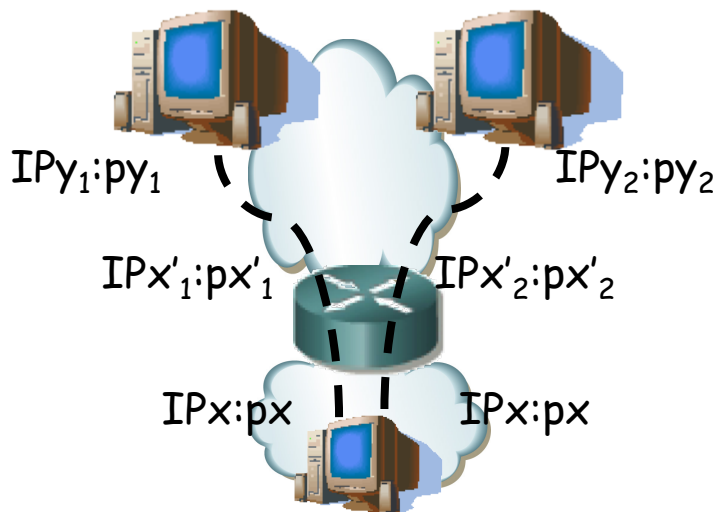
Mapeo de dirección y puerto

- Mapeo establecido: $(IPx:px, IPx'_1:px'_1) \rightarrow IPy_1:py_1$
- A continuación se quiere enviar de $IPx:px$ a $IPy_2:py_2$
- *Endpoint-Independent Mapping*
- *Address-Dependent Mapping*
 - Reutiliza el mapeo establecido con esa dirección y puerto internos siempre que vayan a la misma dirección externa
 - Es decir, $IPx'_1:px'_1 = IPx'_2:px'_2$ si y solo si $IPy'_2 = IPy'_1$



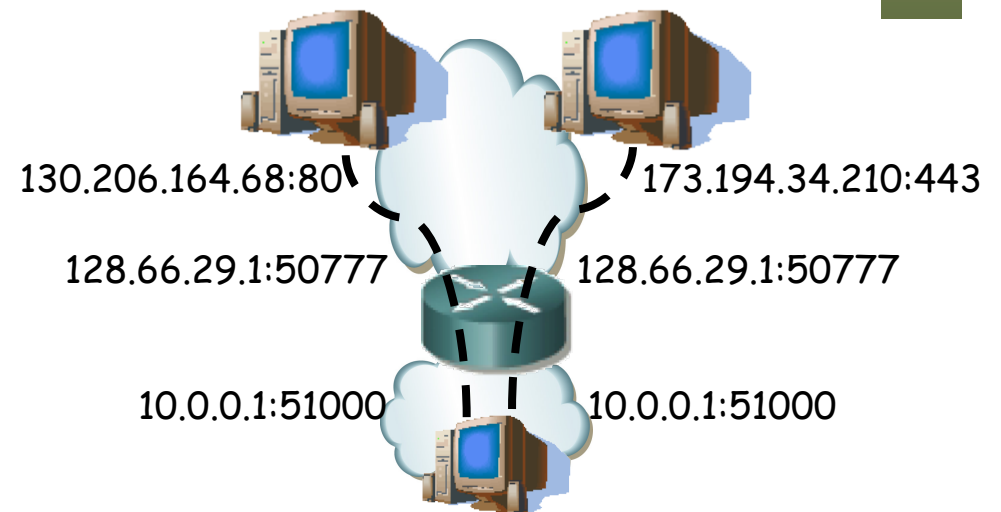
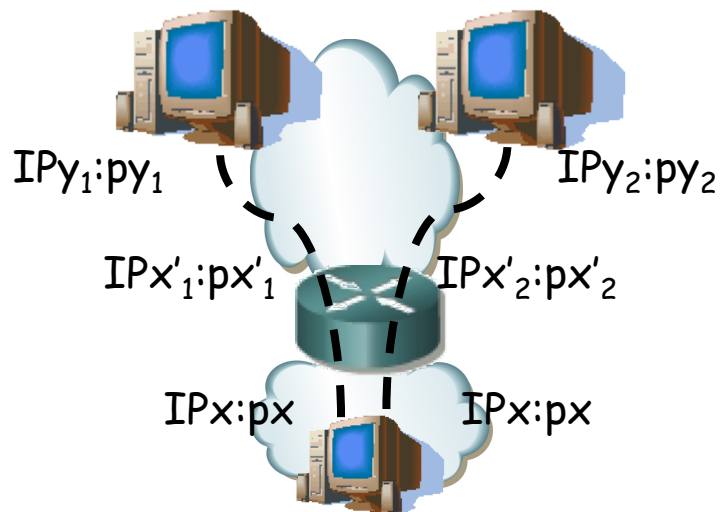
Mapeo de dirección y puerto

- Mapeo establecido: $(IPx:px, IPx'_1:px'_1) \rightarrow IPy_1:py_1$
- A continuación se quiere enviar de $IPx:px$ a $IPy_2:py_2$
- *Endpoint-Independent Mapping*
- *Address-Dependent Mapping*
- *Address and Port-Dependent Mapping*
 - Reutiliza el mapeo establecido con esa dirección y puerto internos siempre que vayan a la misma dirección y puerto externos
 - Es decir, $IPx'_1:px'_1 = IPx'_2:px'_2$ si y solo si $IPy'_2:py_2 = IPy'_1:py_1$



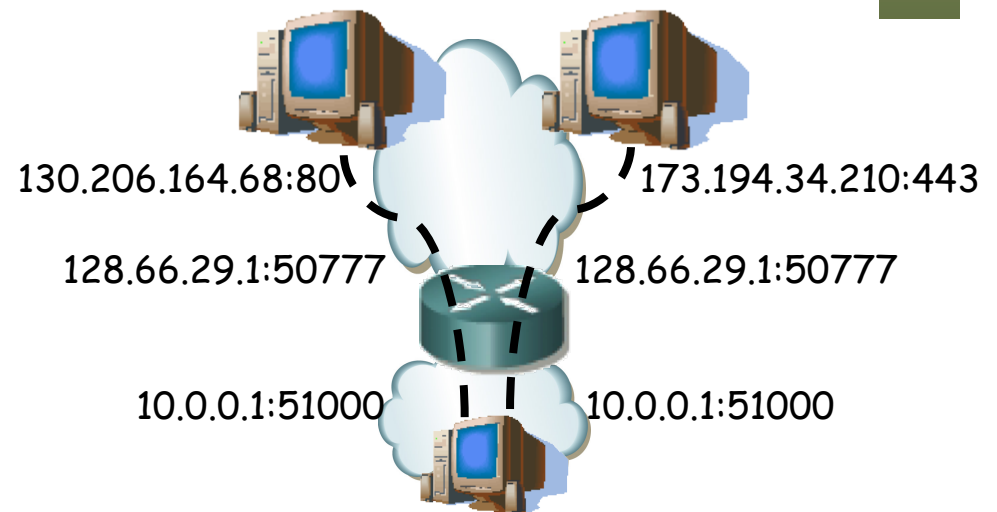
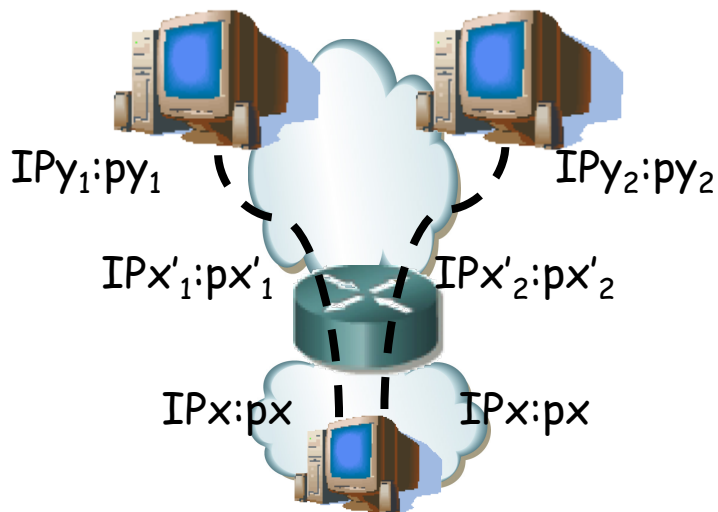
REQ: Endpoint-Independent

- BCP 127 **requiere** comportamiento *Endpoint-Independent*
- Necesario para métodos UNSAF



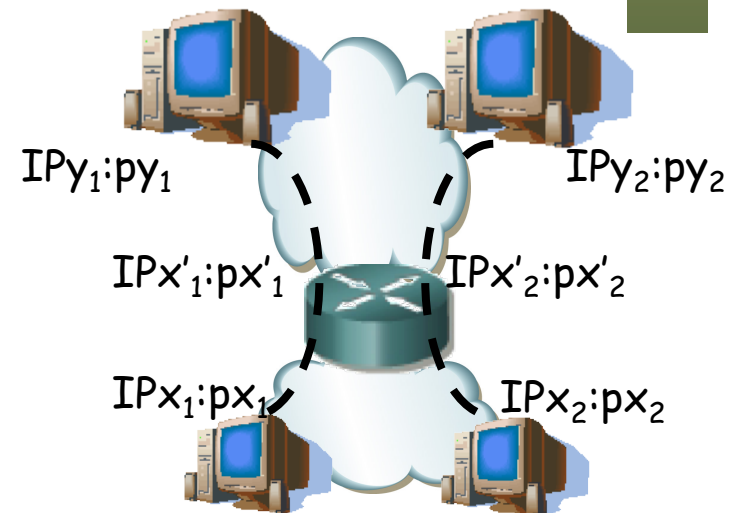
REQ: Emparejado de dir.

- Algunos NATs escogen la dirección pública de un *pool*
- Si lo hacen al azar para cada mapeo no mantienen la identidad (dirección IP) del host externamente
- Suele hacerse para ocultar a los usuarios
- Da problemas con algunas aplicaciones (ej: algunas sobre RTP)
- BCP 127: Requiere que la dirección externa en todos los mapeos de una dirección interna se mantenga



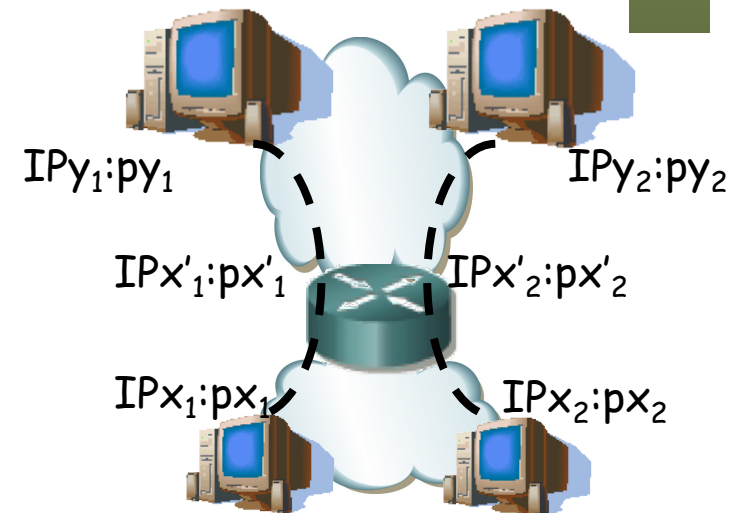
Mapeo de puerto

- *Port preservation*: NAT al mantener puerto: $px_1=px'_1$, $px_2=px'_2$
- Técnicas que usan:
 - Sobrescribir el mapeo anterior
 - Asignar otra dirección pública si tienen un *pool*
 - *Port overloading (PO)*: usar el mismo puerto aunque colisionen
- Puertos: 0-1023 *well-known*, 1024-59151 *registered*, 49152-65535 *dynamic/private*
- Si requiere cambiar el puerto
 - Algunos emplean solo el rango dinámico
 - Otros emplean todos menos los *well-known*
 - Otros mantienen el que sea del mismo grupo si es *well-known*
- REQ BCP 127: No hacer *port overloading*
- Si no dos hosts internos no pueden usar mismo puerto con mismo servidor:puerto
- Recomendado: mantener el puerto en el mismo rango (0-1023 y 1034-65535)



Mapeo de puerto: paridad

- RFC 3550 “RTP: A Transport Protocol for Real-Time Applications”
- “ [...] RTP SHOULD use an even destination port number and the corresponding RTCP stream SHOULD use the next higher (odd) destination port number. For applications that take a single port number as a parameter and derive the RTP and RTCP port pair from that number, if an odd number is supplied then the application **SHOULD** replace that number with the next lower (even) number to use as the base of the port pair. “
- Si el NAT cambia a un puerto impar (odd) y sigue esta regla cambia a otro puerto que el cliente no sabe el que es
- Recomendado: mantener paridad
- RFC 3605 mejora la especificación de puertos para RTP/RTCP

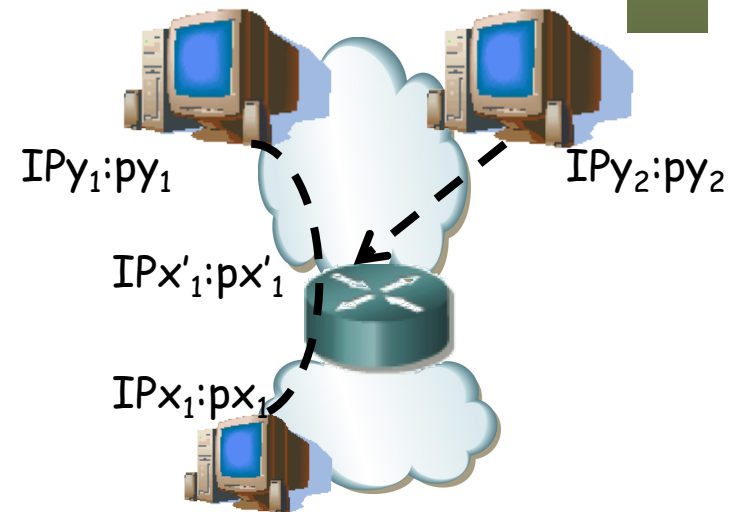


Timers

- Tiempo sin recibir paquetes de un mapeo tras el cual se elimina
- En el rango de unos minutos (2-5min)
- Si es de un servicio (puerto) conocido sobre UDP que se sabe emplea flujos cortos se puede reducir
- REQ: refrescar el timer ante paquetes salientes
- Recomendado: refrescarlo ante paquetes entrantes (desventaja de que el externo puede mantenerlo indefinidamente)

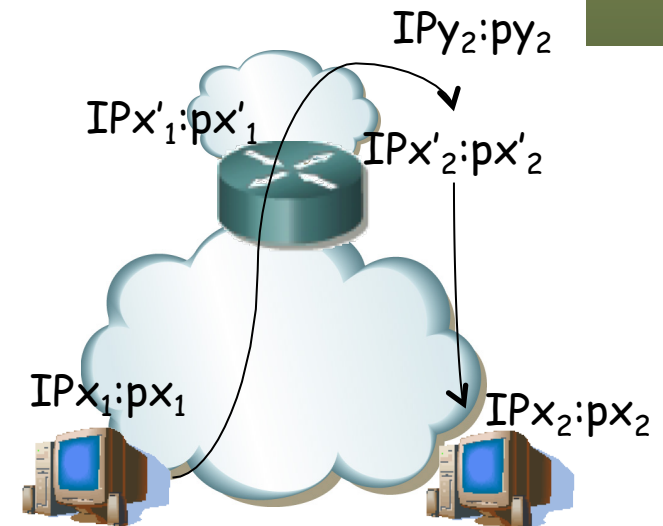
Filtrado

- Mapeo establecido: $(IPx:px, IPx'_1:px'_1) \rightarrow IPy_1:py_1$
- El filtrado determina qué hace el NAT con paquetes de $IPy_2:py_2$ dirigidos a $IPx'_1:px'_1$
- *Endpoint-Independent*
 - Reenvía cualquier paquete dirigido a $IPx_1:px_1$ hacia $IPx_1:px_1$
- *Address-Dependent*
 - Reenvía el paquete hacia $IPx_1:px_1$ solo si $IPy_2=IPx_1$
- *Address and Port-Dependent*
 - Reenvía el paquete hacia $IPx_1:px_1$ solo si $IPy_2=IPx_1$ y $py_2=px'_1$
- Ejemplo: Una aplicación UDP (un juego?) podría cambiar al cliente a otro servidor
- El nuevo servidor interesa que pueda contactar con el cliente si sabe $IPx'_1:px'_1$
- Recomendado:
 - *Endpoint-Independent* para facilitar a las aplicaciones
 - *Address-Dependent* para mayor restricción



Hairpinning

- Dos hosts internos IPx_1 e IPx_2
- Hay un mapeo $(IPx_2:px_2, IPx'_2:px'_2) \rightarrow IPy_2:py_2$
- Es un NAT Endpoint-Independent
- IPx_1 intenta comunicarse con IPx_2 a través de la dirección externa
- Envía paquete $IPx_1:px_1 \rightarrow IPx'_2:px'_2$ (...)
- Se dice que hace *hairpinning* al reenviar ese paquete a $IPx_2:px_2$ (...)
- Comportamiento "*Internal source IP address and port*"
 - El paquete lo reenvía con origen $IPx_1:px_1$
- Comportamiento "*External source IP address and port*"
 - El paquete lo reenvía con origen $IPx'_1:px'_1$
- REQ: Soportar *hairpinning* en modo *External*
- Esto permite que se comuniquen empleando la dirección externa
- Con *Internal* la respuesta a este paquete no pasaría por el NAT y llegaría a IPx_1 desde una dirección inesperada



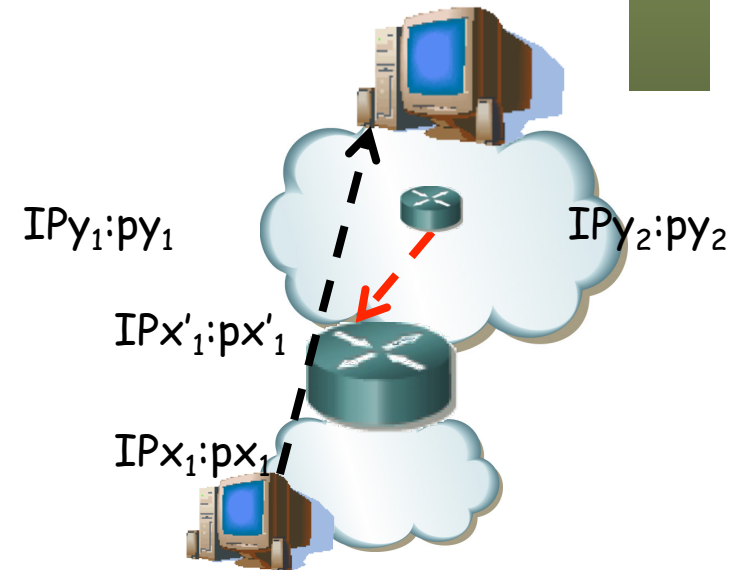
Determinismo

- Hay implementaciones de NAT que cambian de comportamiento según las circunstancias
- Por ejemplo que son *Endpoint-Independent* con *Port Overloading* mientras no haya dos hosts internos en comunicación con el mismo externo y si lo hay cambia a *Address and Port-Dependent* sin *Port preservation*
- Un NAT que cambia su método de mapeo o de filtrado sin cambios de configuración se llama no determinista
- No deterministas suelen cambiar el comportamiento ante conflictos
- Se habla de su comportamiento primario, secundario, terciario, según el número de conflictos
- REQ: comportamiento determinista



ICMP destino inalcanzable

- Puede ser devuelto por un router intermedio
- Dirección IP origen del ICMP no coincide con la del mapeo
- No debería descartarlo
- Debería aplicarle el mapeo y reescribir los datos (el paquete contenido tiene la dirección pública, no la interna)
- Debe hacerlo mientras exista el mapeo
- REQ: No debe eliminar el mapeo por recibir un ICMP
- Permite así UDP Path MTU Discovery y traceroute



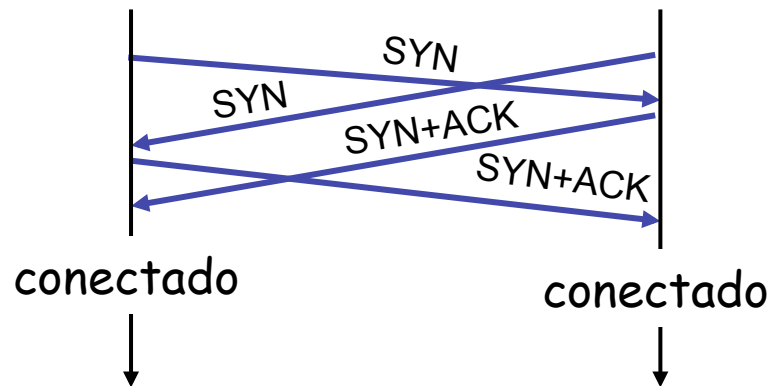
Recibiendo fragmentos

- Si recibe fragmentos del exterior la cabecera de transporte está solo en el primero (que puede no llegar el primero)
- NAT “*Received Fragments Ordered*”
 - Solo es capaz de reenviar correctamente los fragmentos si han llegado en orden
- NAT “*Received Fragments Out of Order*”
 - Es capaz de reenviar fragmentos correctamente aunque le lleguen desordenados
- REQ: ser *out of order*



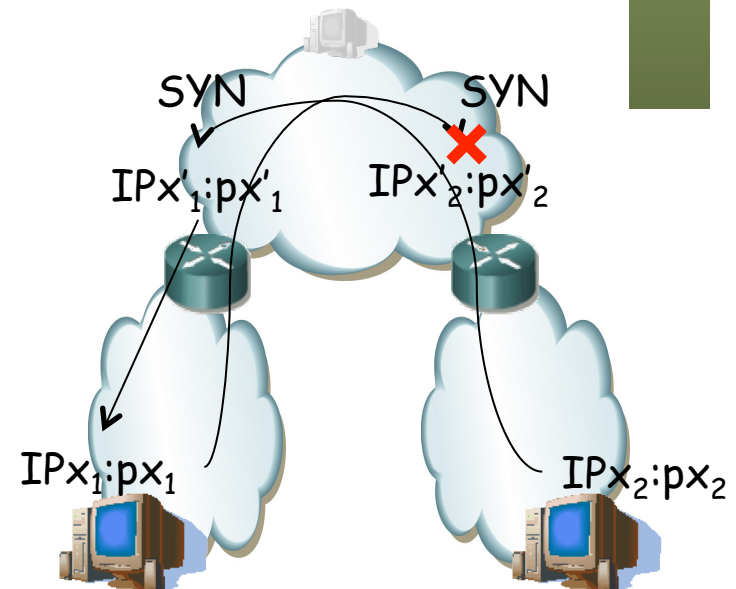
REQs TCP: RFC 5382

- BCP 143 “NAT Behavioral Requirements for TCP” (2008)
- REQ: Comportamiento “*Endpoint-Independent Mapping*”
 - Es decir, mantener mismos IP:puerto externos para mismos internos independiente del destino externo
 - Necesario para UNSAF
- REQ: Debe soportar todas las secuencias de paquetes TCP; en concreto debe manejar correctamente un *simultaneous open*
 - Que no impida un SYN entrante (sin ACK) si hay mapeo
 - Permite hacer *hole punching* para comunicar dos hosts ambos tras NATs (...)



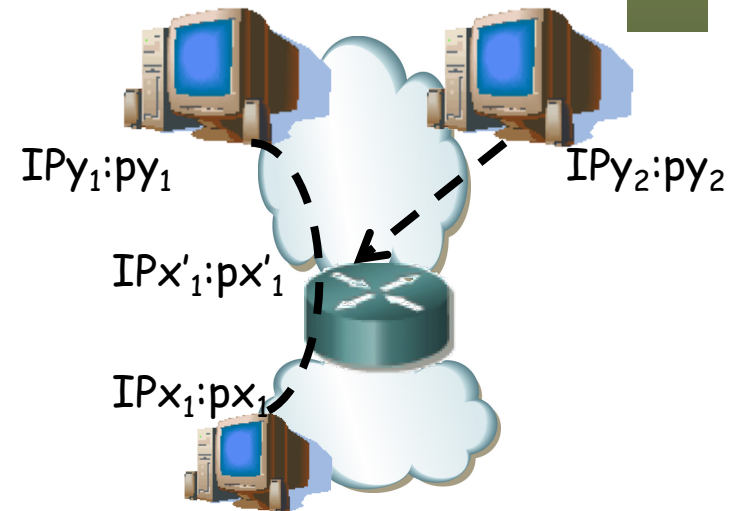
Hole Punching

- Partimos de que cada extremo conoce la dirección y puerto externo del otro
- Por ejemplo cada uno ha hablado con un servidor externo, de forma que éste ha visto el mapeo externo
- Si el NAT hace un *Endpoint-Independent Mapping* mantendrá esas IP:puertos externos siempre que el cliente mantenga los suyos, aunque vayan los paquetes a otro host
- Host 1 manda SYN a $IPx'_2:px'_2$ mientras host 2 manda SYN a $IPx'_1:px'_1$
- El primero podría no encontrará mapeo (...)
- El segundo sí (...)
- Host 1 ha enviado SYN y recibido otro
- ¡ *Simultaneous open* !
- Si al primer paquete el NAT contesta con un ICMP abortaría el proceso
- REQ: espera 6s para ese ICMP
- Si en ese tiempo crea un mapeo saliente descarta el SYN entrante viejo



BCP 143

- Recomendado (como para UDP):
 - *Endpoint-Independent* para facilitar a las aplicaciones (cualquier host externo puede enviar a IP:puerto externo)
 - *Address-Dependent* para mayor restricción (solo el mismo host)



BCP 143: *Session Refresh*

- El mantenimiento de estado es sensible a ataques y agotamiento de recursos
- Además una conexión TCP puede permanecer establecida indefinidamente sin enviar tráfico
- NAT podría eliminar mapeos de sesiones que estima han terminado en el orden:
 - Sesiones en que un extremo se ha reiniciado
 - Sesiones cerradas
 - Sesiones en proceso de establecerse
- El NAT podría comprobar si un extremo se ha reiniciado enviándole un *keep-alive* (recibiría un RST)
- REQ: si no puede determinar si los extremos siguen activos puede abandonar la sesión tras un tiempo
- Ese tiempo debe ser al menos de 2h04m para las establecidas



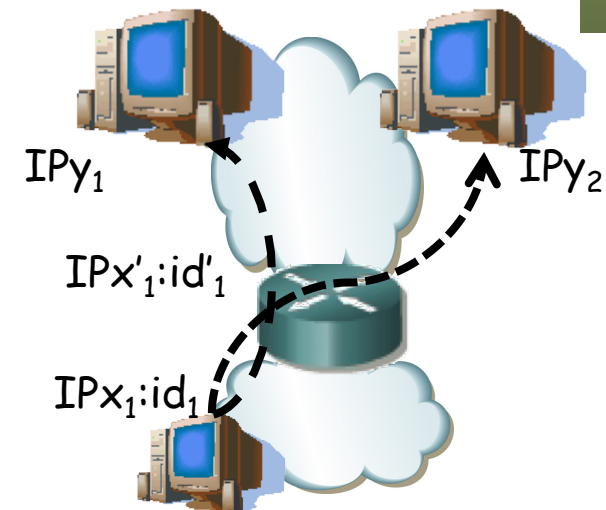
BCP 143

- REQ: no debe usar “*Port overloading*”
- REQ: debe soportar *hairpinning* con comportamiento “*External source IP address and port*” (comunicación de hosts internos a través de direcciones externas)
- REQ: debe traducir ICMP de tipo destino inalcanzable
- REQ: recibir un ICMP no debe eliminar un mapeo

REQs ICMP: RFC 5508

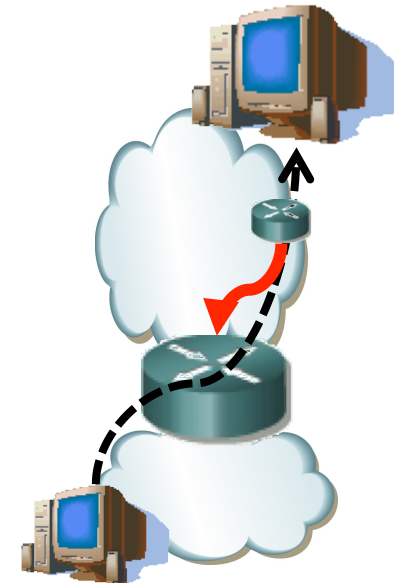
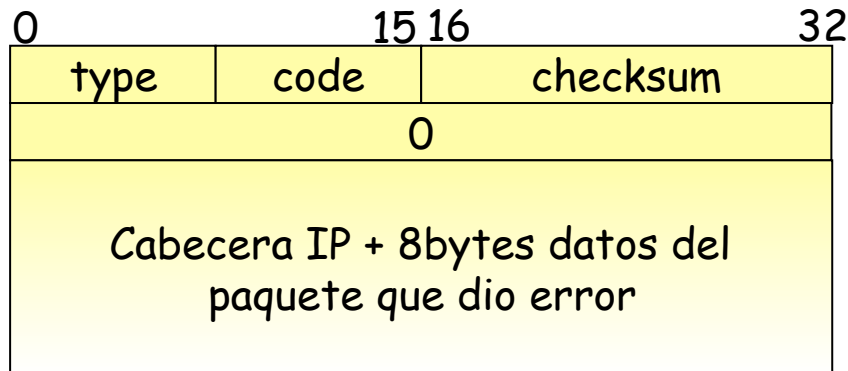
- BCP 148: “NAT Behavioral Requirements for ICMP” (2009)
- REQ: debe permitir *ICMP Queries* desde el interior y sus respuestas. Los identificadores de query externos deberían ser independientes del host externo
 - Es decir, dos queries a distinto destino pero con el mismo identificador en el interior deberían tener el mismo en el exterior
- REQ: el timeout de la sesión debe ser al menos de 60s
- REQ: el NAT debe descartar errores ICMP cuyo checksum ICMP no sea correcto

0		15 16		32
type	code	checksum		
Identificador		Número de secuencia		
Datos (opcional)				



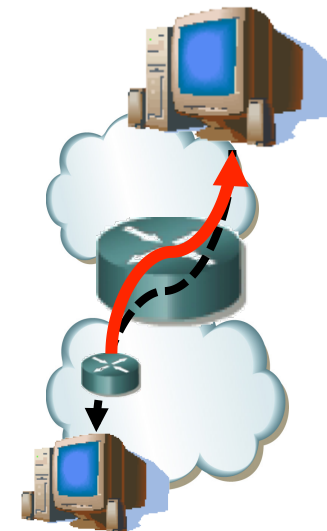
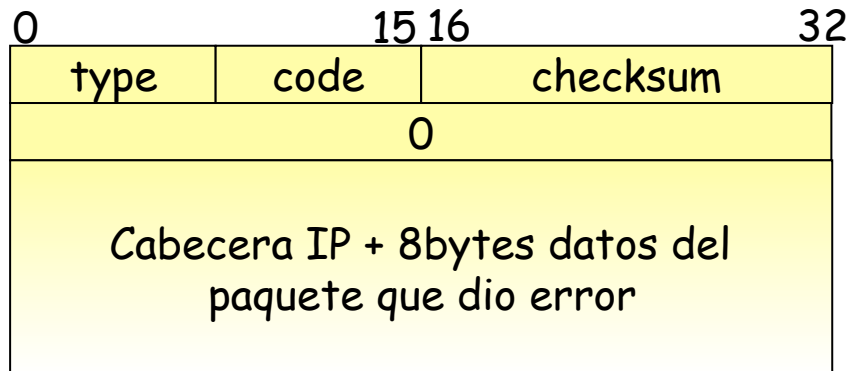
BCP 148

- REQ: si el NAT recibe un ICMP Error desde el exterior y tiene un mapeo activo para el paquete contenido debe:
 - Mapear la cabecera IP y de transporte del paquete contenido
 - Modificar la dirección IP destino del ICMP a ser la dirección origen del paquete contenido
- Si no tiene mapeo debería descartar el paquete ICMP



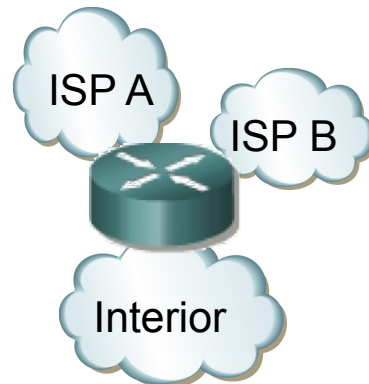
BCP 148

- REQ: si el NAT recibe un ICMP Error desde el **interior** y tiene un mapeo activo para el paquete contenido debe:
 - Mapear la cabecera IP y de transporte del paquete contenido
 - Si tiene mapeo para la dirección origen del error traducirla y si no tiene cambiarla por su propia dirección IP pública
- Si no tiene mapeo debería descartar el paquete ICMP
- REQ: debe soportar *hairpinning*



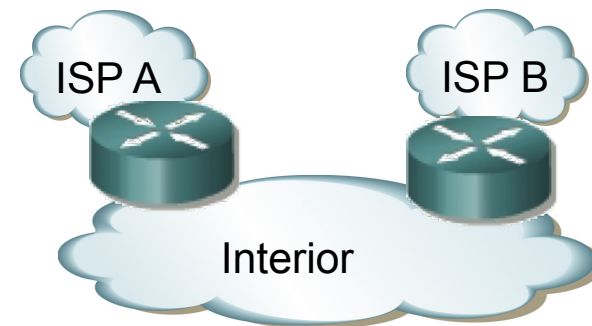
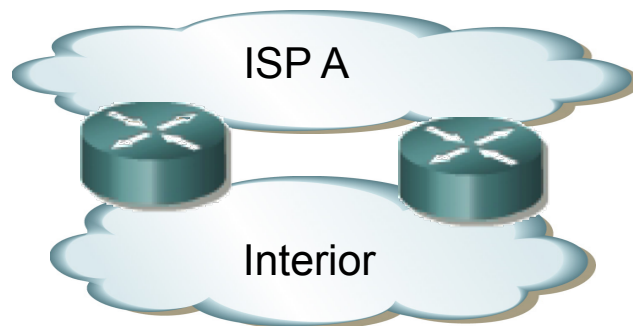
Protección y NATs

- Mantiene el mapeo para una sesión así que todo el tráfico debe pasar por él
- No debe haber asimetría, los dos sentidos de la comunicación deben pasar por el NAT
- Normalmente el NAT está en la frontera de un *stub*
- Es un punto único de fallo
- ¿ Multihomed ?
 - Protección con enlaces a varios operadores
 - La dirección externa será diferente así que los mapeos fallarán salvo que se enrute de vuelta al NAT por el segundo operador



Protección y NATs

- Mantiene el mapeo para una sesión así que todo el tráfico debe pasar por él
- No debe haber asimetría, los dos sentidos de la comunicación deben pasar por el NAT
- Normalmente el NAT está en la frontera de un *stub*
- Es un punto único de fallo
- ¿ Múltiples NATs ?
 - Protección ante fallo del equipo y de enlace si van a diferente ISP
 - Podría emplearse para cada sesión el más cercano al destino
 - Si uno falla debería encaminarse el tráfico por el otro
 - Las sesiones entonces deben estar sincronizadas
 - De nuevo problema con la dirección externa

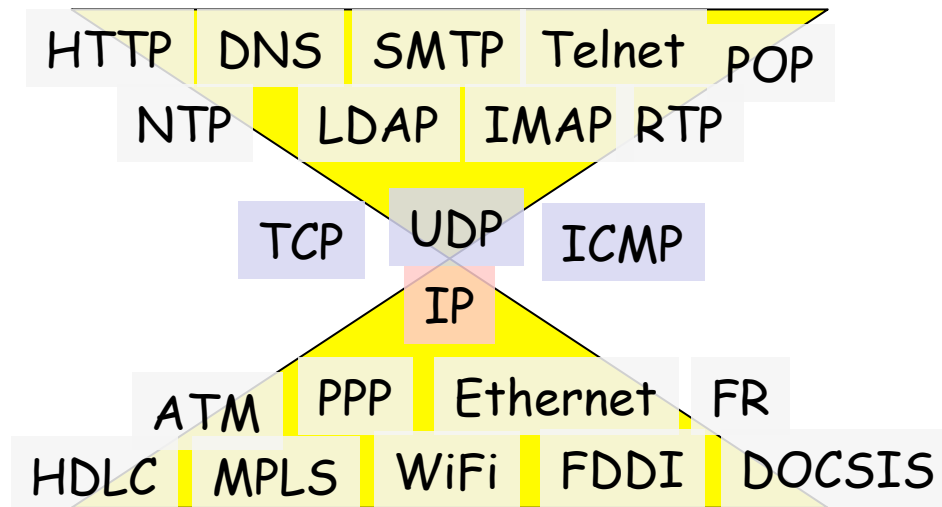


Seguridad y NATs

- Hosts internos no son directamente accesibles si no inician ellos la comunicación
- Cambian direcciones y puertos así que no funcionan con mecanismos de seguridad que los firmen (IPSec)
- Sí funcionan con aplicaciones que no basen la seguridad en direcciones IP o puertos (SSH, TLS)

Protocolos sobre IP

- Campo protocolo 1 byte (256 valores)
- Hay ya 142 reservados (2013)
- Pero un NAT soporta traslación solo para TCP/UDP/ICMP
- Con el despliegue que hay de NATs, el empleo de otros protocolos hoy en día no tendría alcance global
- Nuevos protocolos acaban implementándose sobre UDP que da un servicio de datagramas como IP



Resumen

- NATs modifican cabecera de red y transporte
- Necesarios debido al agotamiento de direcciones
- También dan cierta seguridad añadida
- Dan problemas serios a las aplicaciones
- Los desarrolladores de aplicaciones deben tenerlos muy en cuenta