

# NATs: Recomendaciones

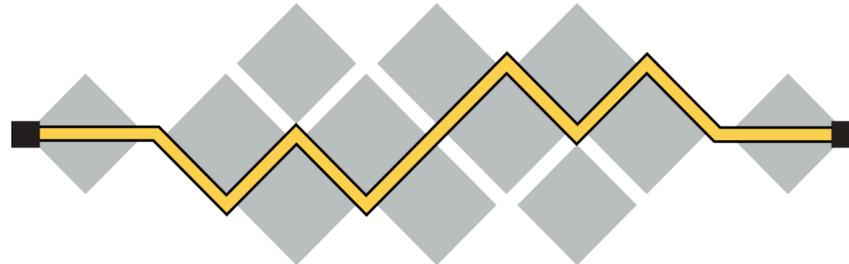
Area de Ingeniería Telemática  
<http://www.tlm.unavarra.es>

Grado en Ingeniería en Tecnologías de  
Telecomunicación, 3º

# NATs y UDP: Mapeos

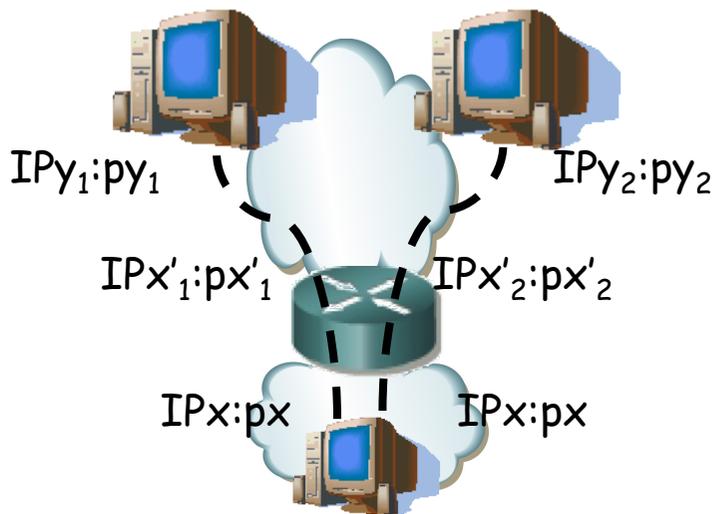
# REQs UDP: RFC 4787

- 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 (...)



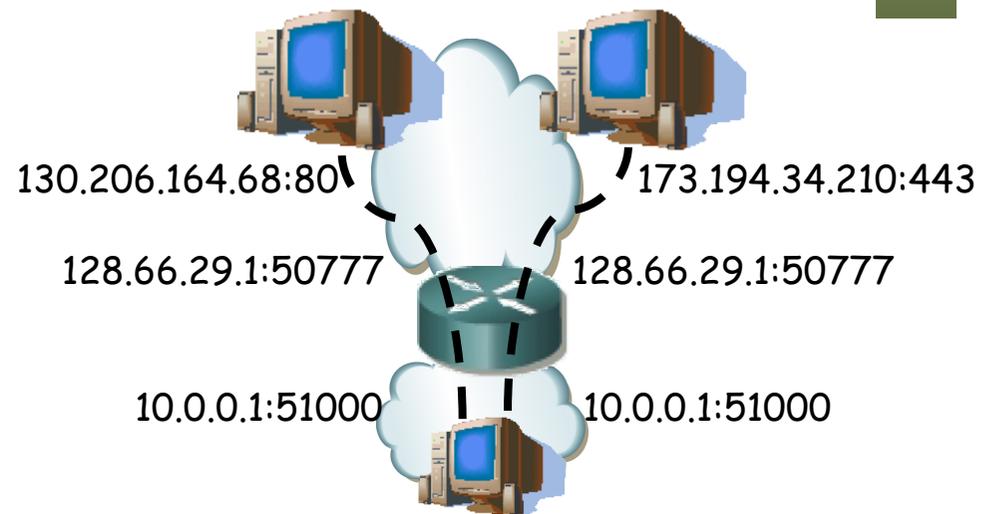
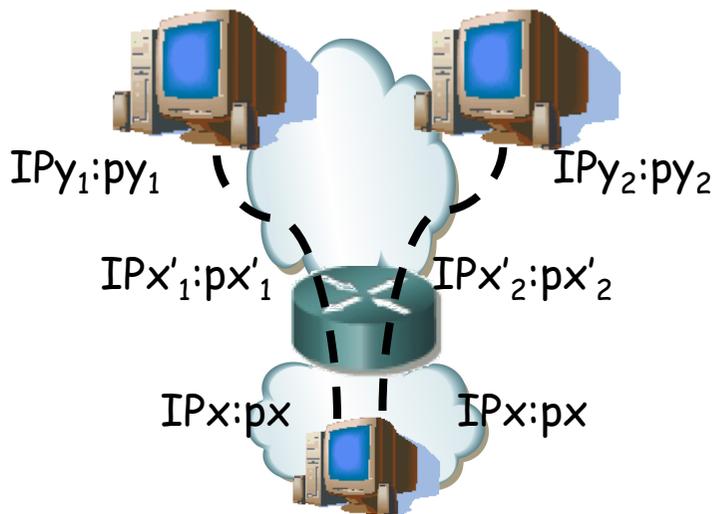
# Mapeo de dirección y puerto

- 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
- (...)



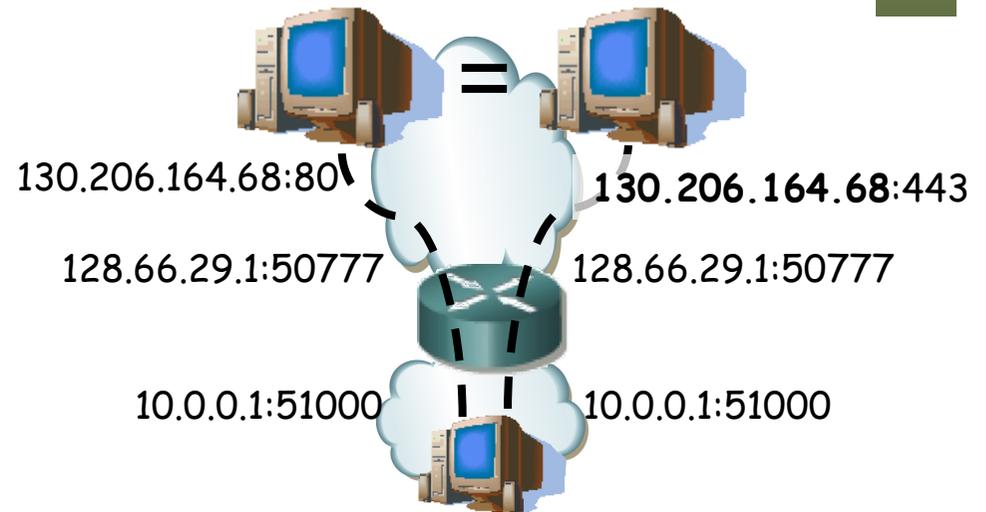
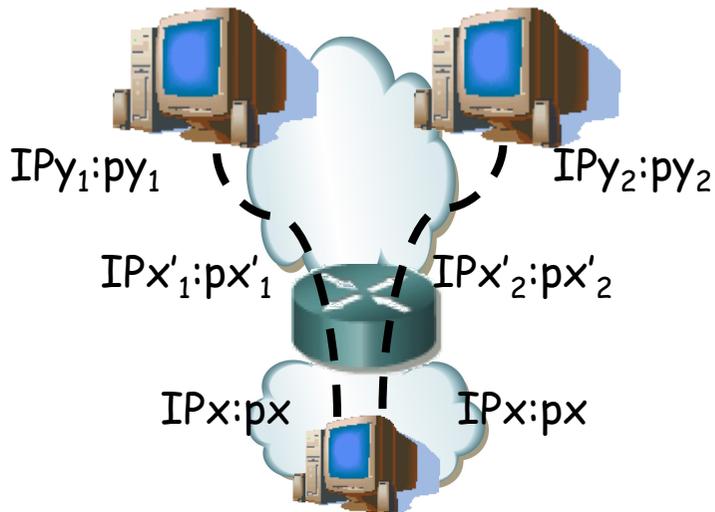
# Mapeo de dirección y puerto

- 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  $IPy'_2:py'_2$
  - (...)



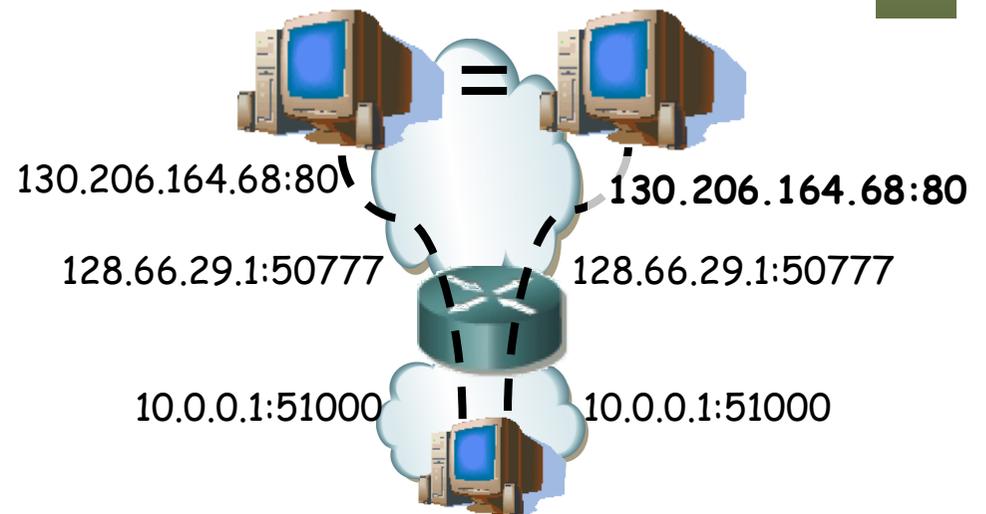
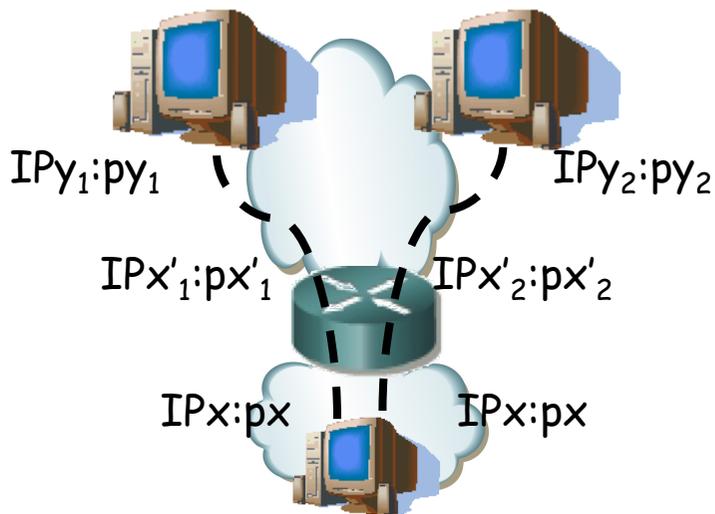
# Mapeo de dirección y puerto

- 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  $IPy'_2 = IPy'_1$
  - (...)



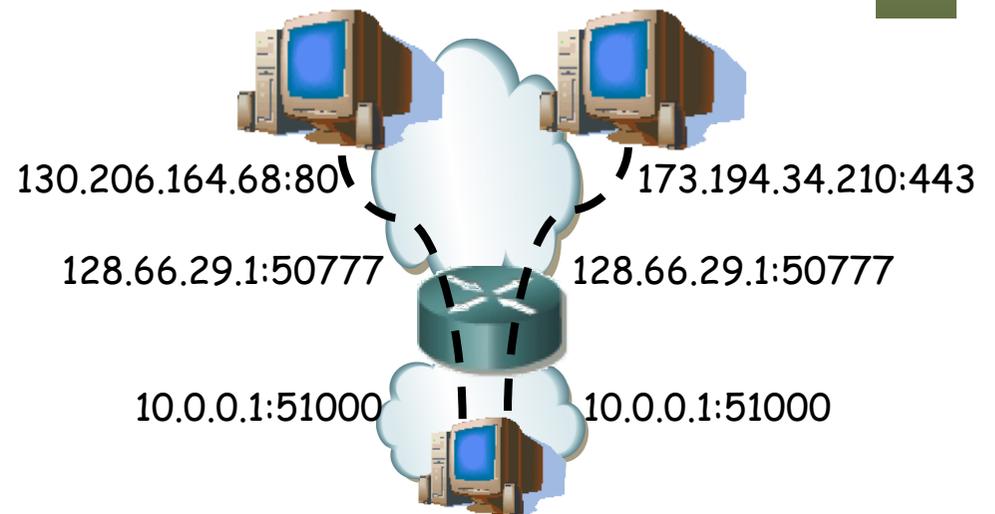
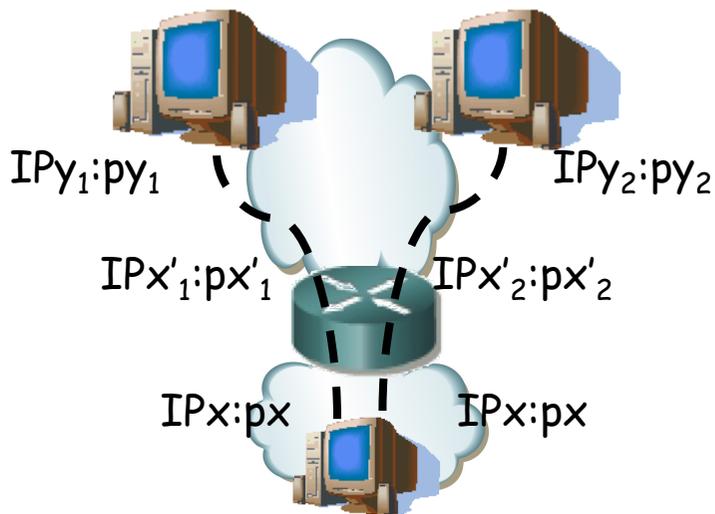
# Mapeo de dirección y puerto

- 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  $IPy'_2:py_2 = IPy'_1:py_1$



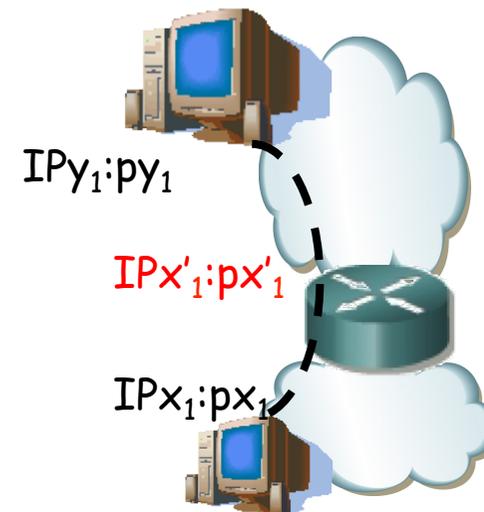
# REQ: Endpoint-Independent

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



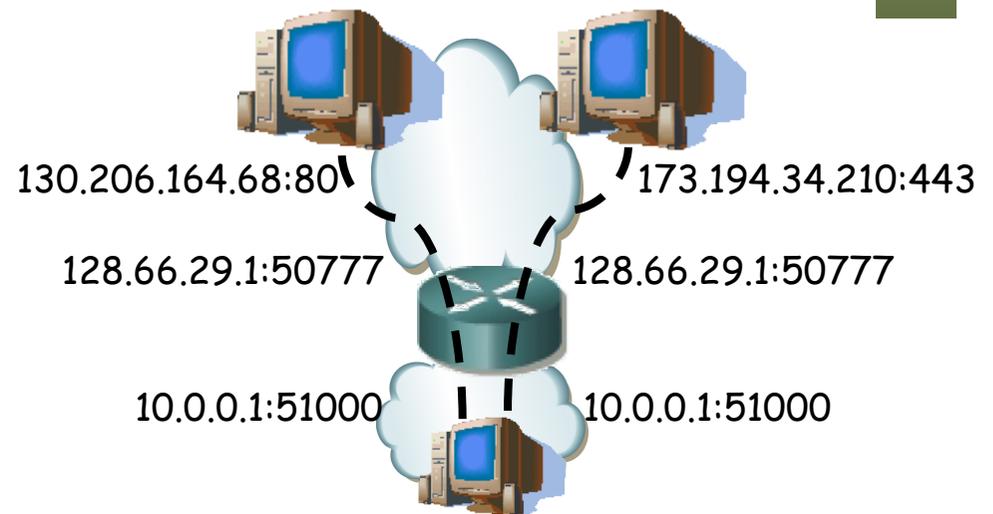
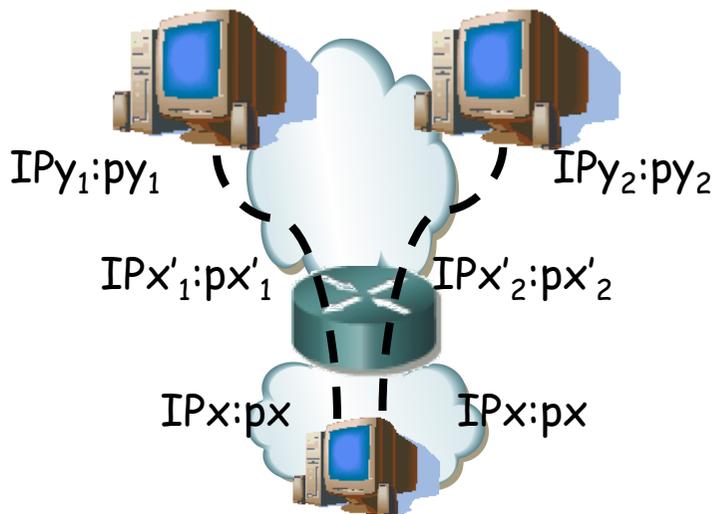
# UNSAF

- 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



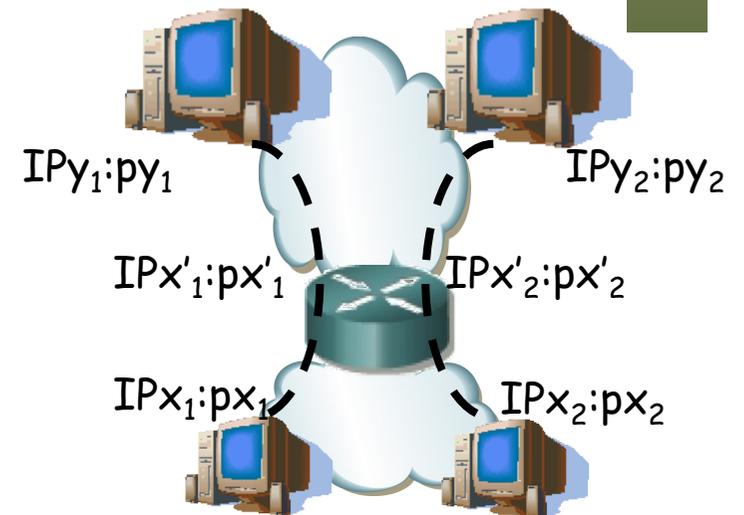
# REQ: Emparejado de dir.

- 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



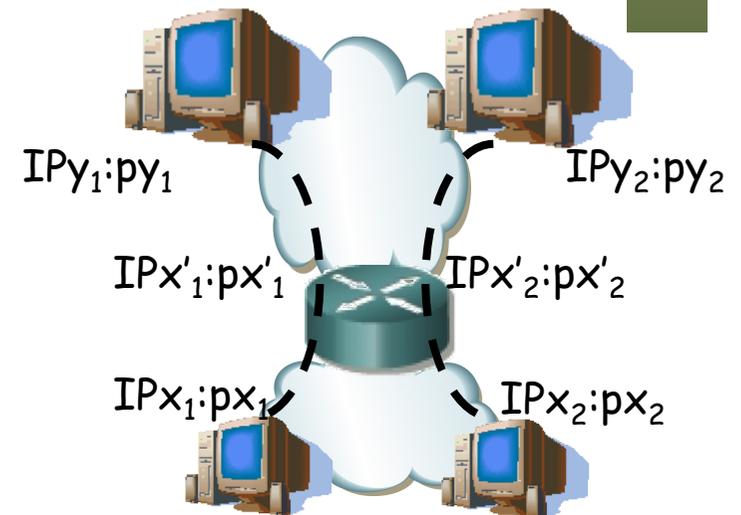
# Mapeo de puerto

- *Port preservation*: NAT al mapear 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
- (...)



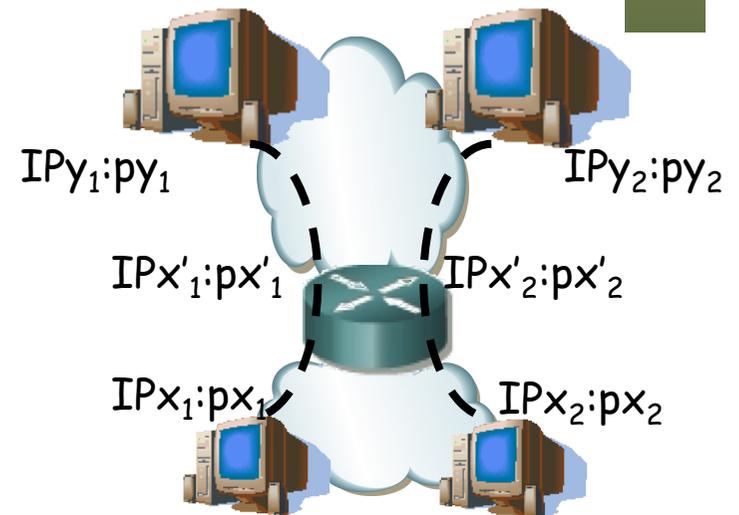
# Mapeo de puerto

- *Port preservation*: NAT al mapear 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*, 1024-49151 *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*
- (...)



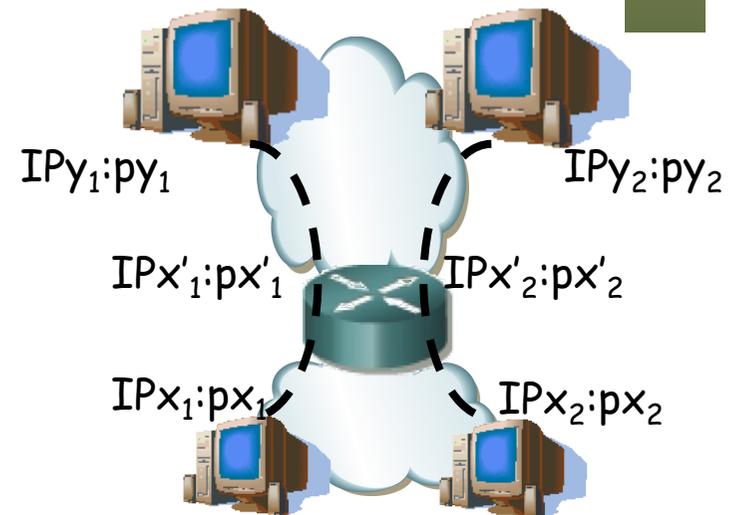
# Mapeo de puerto

- *Port preservation*: NAT al mapear 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*, 1024-49151 *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 1024-65535)



# 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



# NATs y UDP: Timers

# Timers

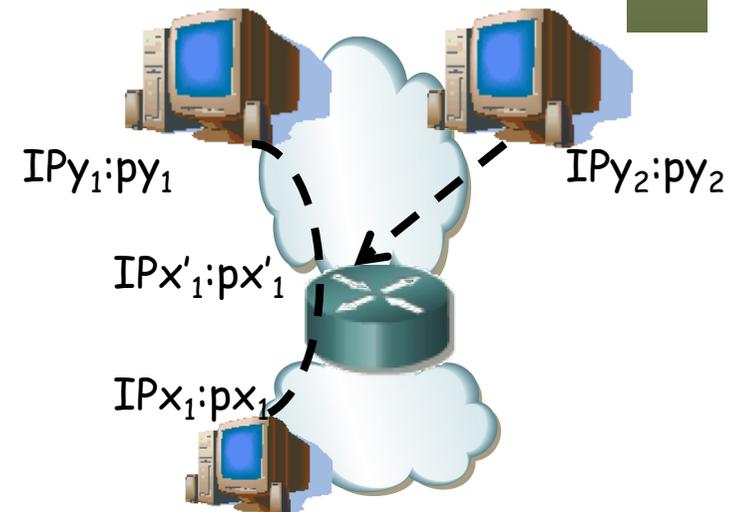
- Tiempo sin recibir paquetes de un mapeo tras el cual se elimina
- En el rango de unos minutos (2-5min)
- Si es de un servicio (puerto) conocido sobre UDP que se sabe emplea flujos cortos se puede reducir
- REQ: refrescar el timer ante paquetes salientes
- Recomendado: refrescarlo ante paquetes entrantes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



# NATs y UDP: Filtrado

# Filtrado

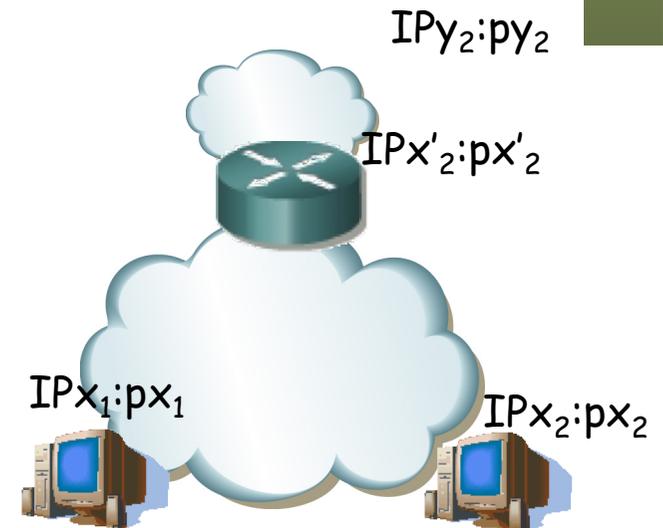
- Mapeo establecido:  $(IPx_1:px_1, IPx'_1:px'_1) \rightarrow IPy_1:py_1$
- El filtrado determina qué hace el NAT con paquetes de  $IPy_2:py_2$  dirigidos a  $IPx'_1:px'_1$
- *Endpoint-Independent*
  - Reenvía cualquier paquete dirigido a  $IPx_1:px_1$  hacia  $IPx_1:px_1$
- *Address-Dependent*
  - Reenvía el paquete hacia  $IPx_1:px_1$  solo si  $IPy_2=IPy_1$
- *Address and Port-Dependent*
  - Reenvía el paquete hacia  $IPx_1:px_1$  solo si  $IPy_2=IPy_1$  y  $py_2=py_1$
- 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



# NATs y UDP: Hairpinning

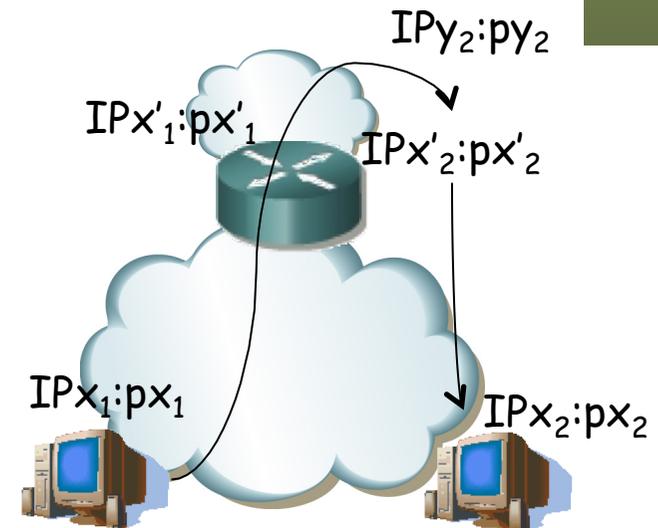
# Hairpinning

- 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  $IPx_2:px_2$



# 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



# NATs y UDP: Otros aspectos

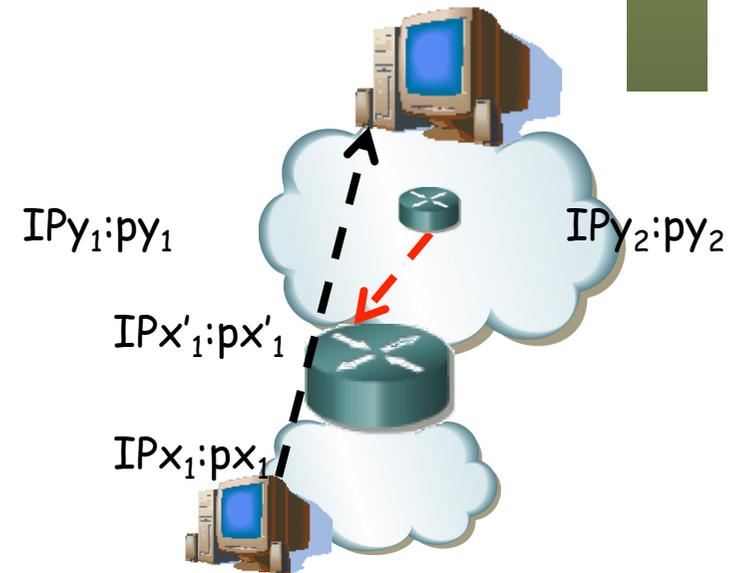
# Determinismo

- Hay implementaciones de NAT que cambian de comportamiento según las circunstancias
- Por ejemplo:
  - Es *Endpoint-Independent* con *Port Overloading*
  - Pero si hay dos hosts internos en comunicación con el mismo externo 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

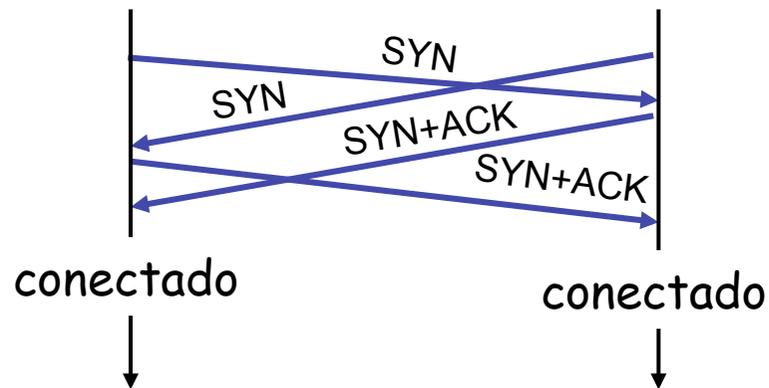
- Si recibe fragmentos del exterior la cabecera de transporte está solo en el primero (que puede no llegar el primero)
- NAT “*Received Fragments Ordered*”
  - Solo es capaz de reenviar correctamente los fragmentos si han llegado en orden
- NAT “*Received Fragments Out of Order*”
  - Es capaz de reenviar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



# NATs y TCP

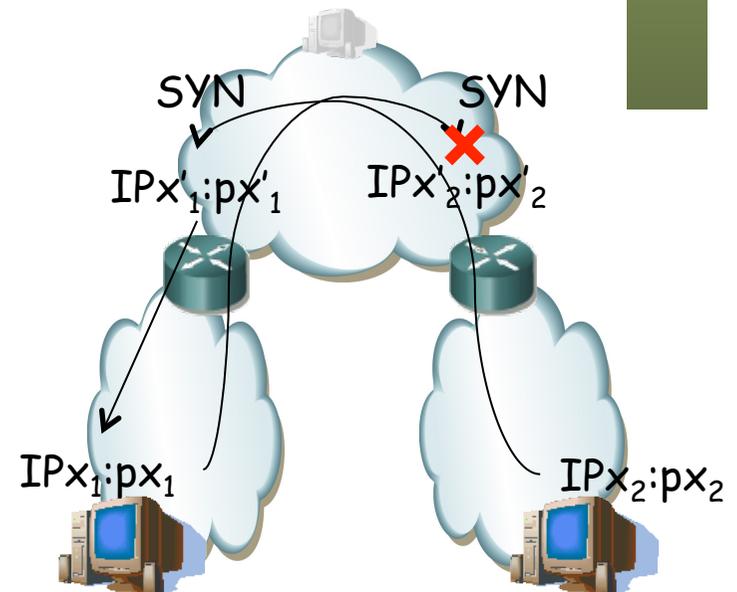
# REQs TCP: RFC 5382

- 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 (...)



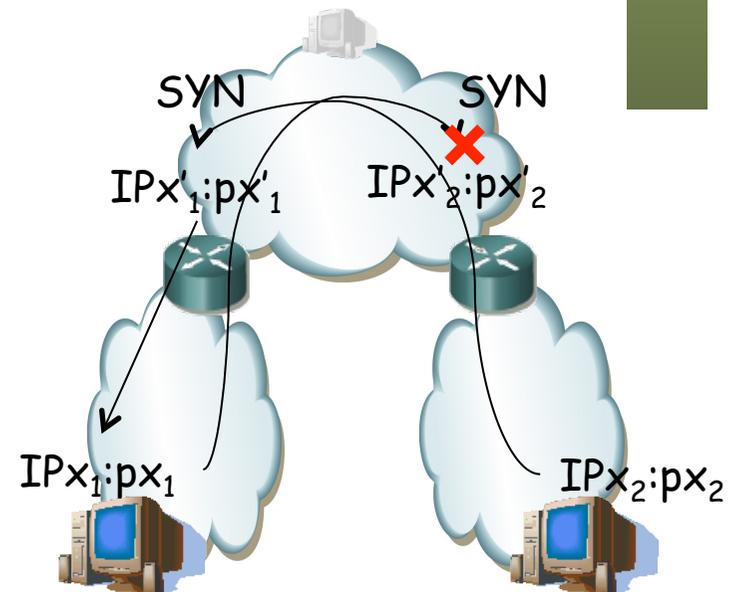
# Hole Punching

- 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
- 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 (...)
- (...)



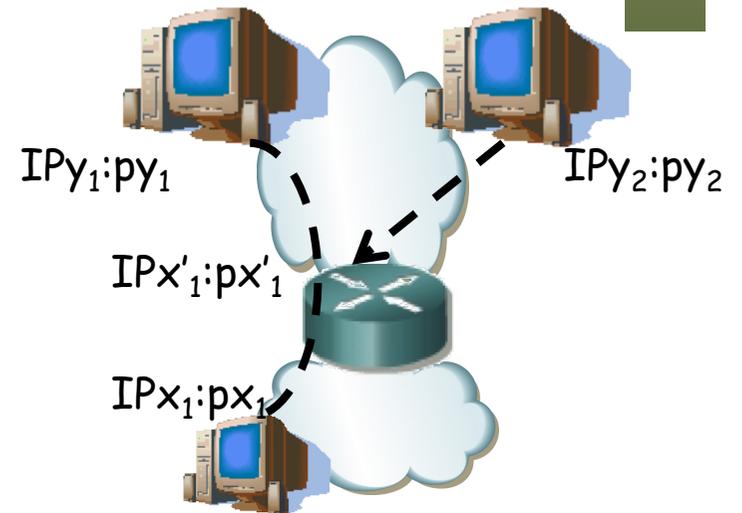
# Hole Punching

- 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
- 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