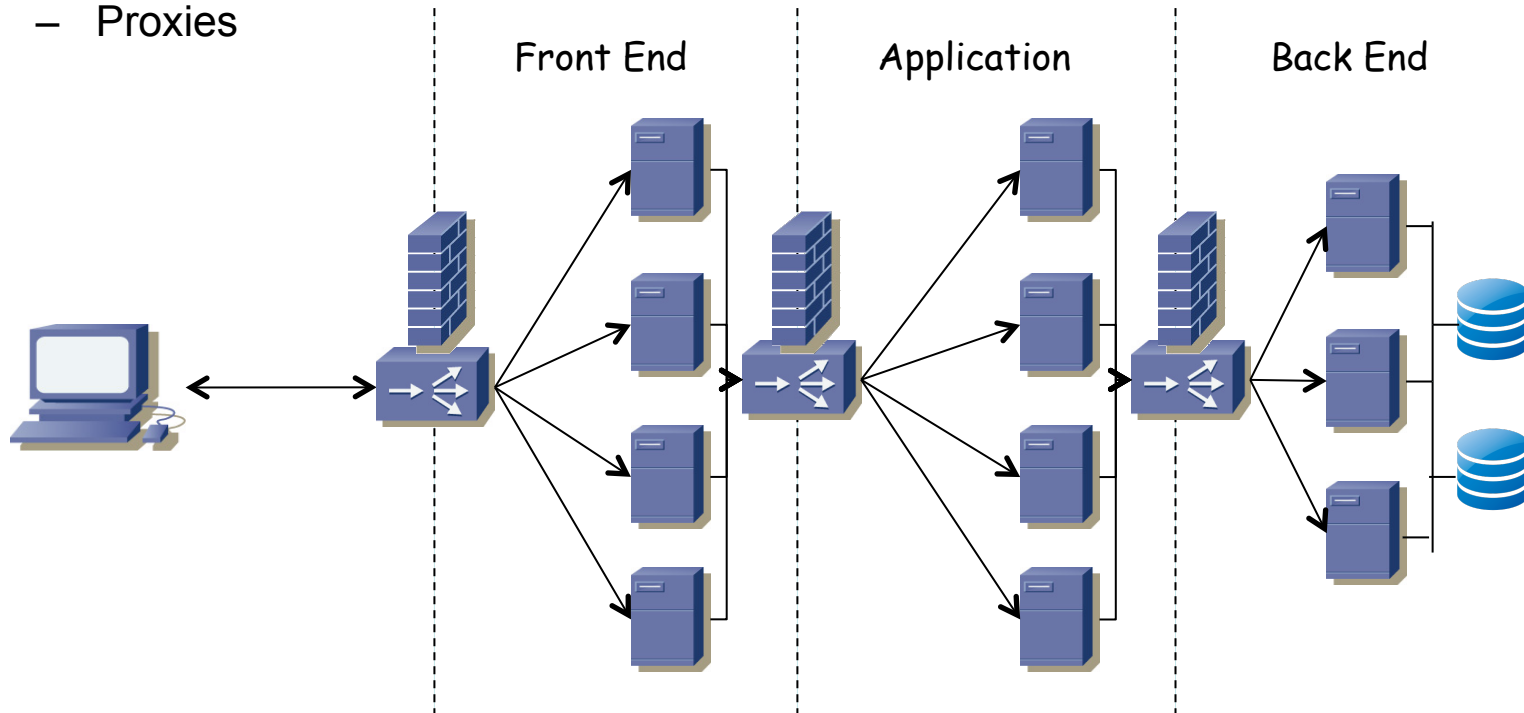


Balanceadores

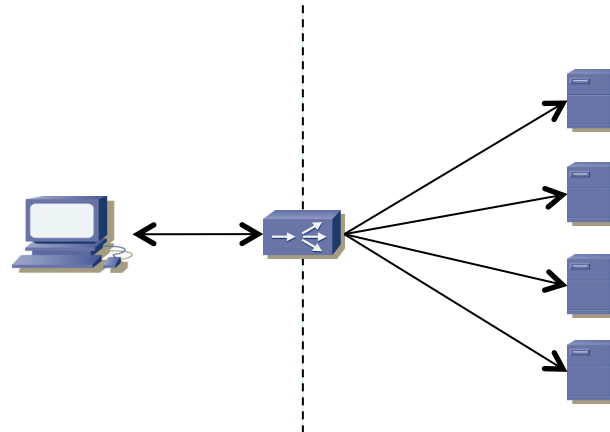
Servicios y multitier

- Hemos visto que es común la separación en capas del servicio
- Entre ellas nos podremos encontrar diferentes servicios:
 - Balanceadores de carga (*content switching*)
 - Firewalls
 - IDSs (Intrusion Detection Systems)
 - SSL Offloading
 - Caches
 - Proxies



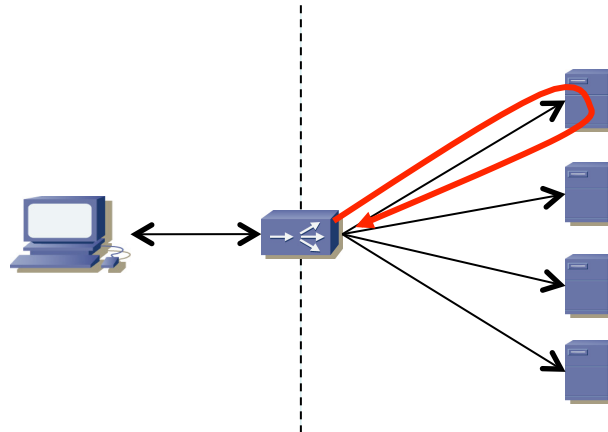
Balanceo de carga

- Antes se hacía cambiando la respuesta DNS
- Hoy en día además con balanceadores (*appliances* o módulos)
- Reparten carga entre los servidores repartiendo las peticiones



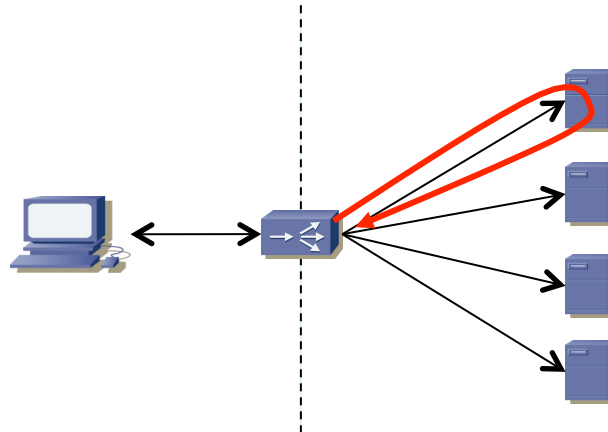
Balanceadores: *health tracking*

- Hacen un seguimiento del estado de los servidores/el servicio (*server health tracking*)
- *In-band*
 - Balanceador monitoriza el tráfico del servidor
 - Esto comprueba que está activo pero no que el servicio se comporta correctamente
 - No requiere que el balanceador entienda el servicio
 - Por ejemplo le reenvía el SYN y si no contesta puede enviar la retransmisión del SYN a otro servidor
 - Si falla numerosas veces puede retirarlo del servicio balanceado
 - Puede monitorizar los códigos de respuesta (por ejemplo HTTP)
- (...)



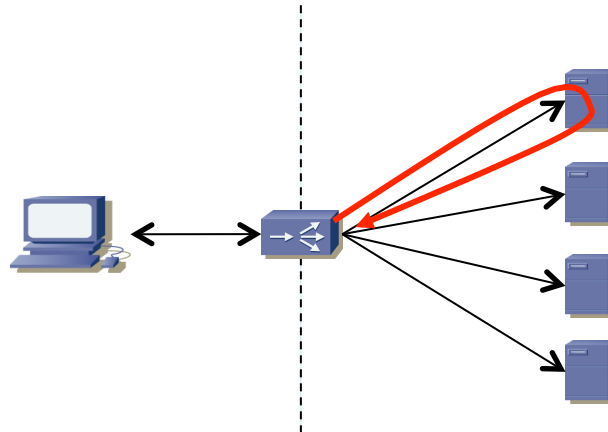
Balanceadores: *health tracking*

- Hacen un seguimiento del estado de los servidores/el servicio (*server health tracking*)
- *In-band*
- *Out-of-band*
 - Balanceador sondea activamente al servidor
 - Puede comprobar solo que la máquina está activa (*server availability*)
 - Puede comprobar que la aplicación está activa (*application availability*)
 - Puede pedir recursos de la aplicación para verificar que entrega lo correcto (*application consistency*)
 - (...)



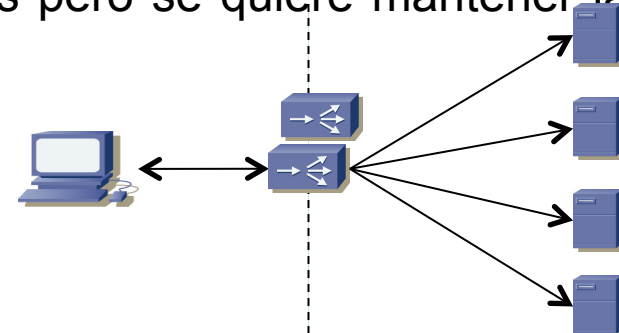
Balanceadores: *health tracking*

- Hacen un seguimiento del estado de los servidores/el servicio (*server health tracking*)
- *In-band*
- *Out-of-band*
 - Sondeo en capa 2: ARP requests
 - Sondeo en capa 3: ICMP echo request
 - Sondeo en capa 4: Establecimiento de conexión TCP
 - Sondeo en capa 5+: Según la aplicación, por ejemplo enviar una petición HTTP y comprobar la respuesta si es la esperada
 - Sondeo SNMP: Puede obtener mucha información según las MIBs soportadas



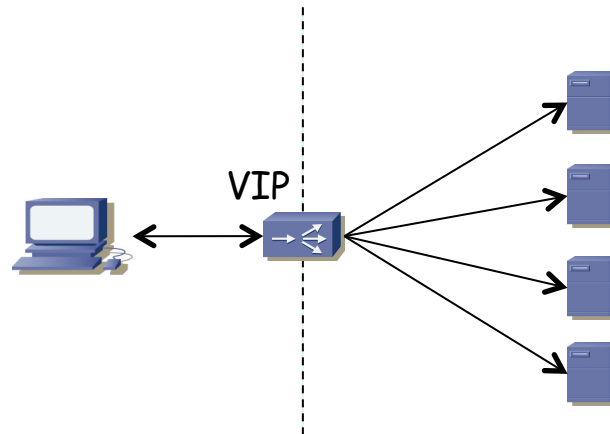
Balanceadores y *failover*

- *Stateless failover*
 - Pasa a usarse el de respaldo al fallar el principal
 - No se mantiene el estado de las sesiones
 - Es útil cuando no hay sesión y las conexiones son muy cortas
- *Stateful failover*
 - Replica el estado como para seguir mandando las conexiones al mismo servidor
 - Útil para conexiones de larga duración
- *Sticky failover*
 - Replica en el de respaldo suficiente estado para seguir reenviando al cliente al mismo servidor
 - Eso permite mantener la sesión aunque rompa la conexión
 - Útil cuando las conexiones son cortas pero se quiere mantener la sesión



Balanceadores

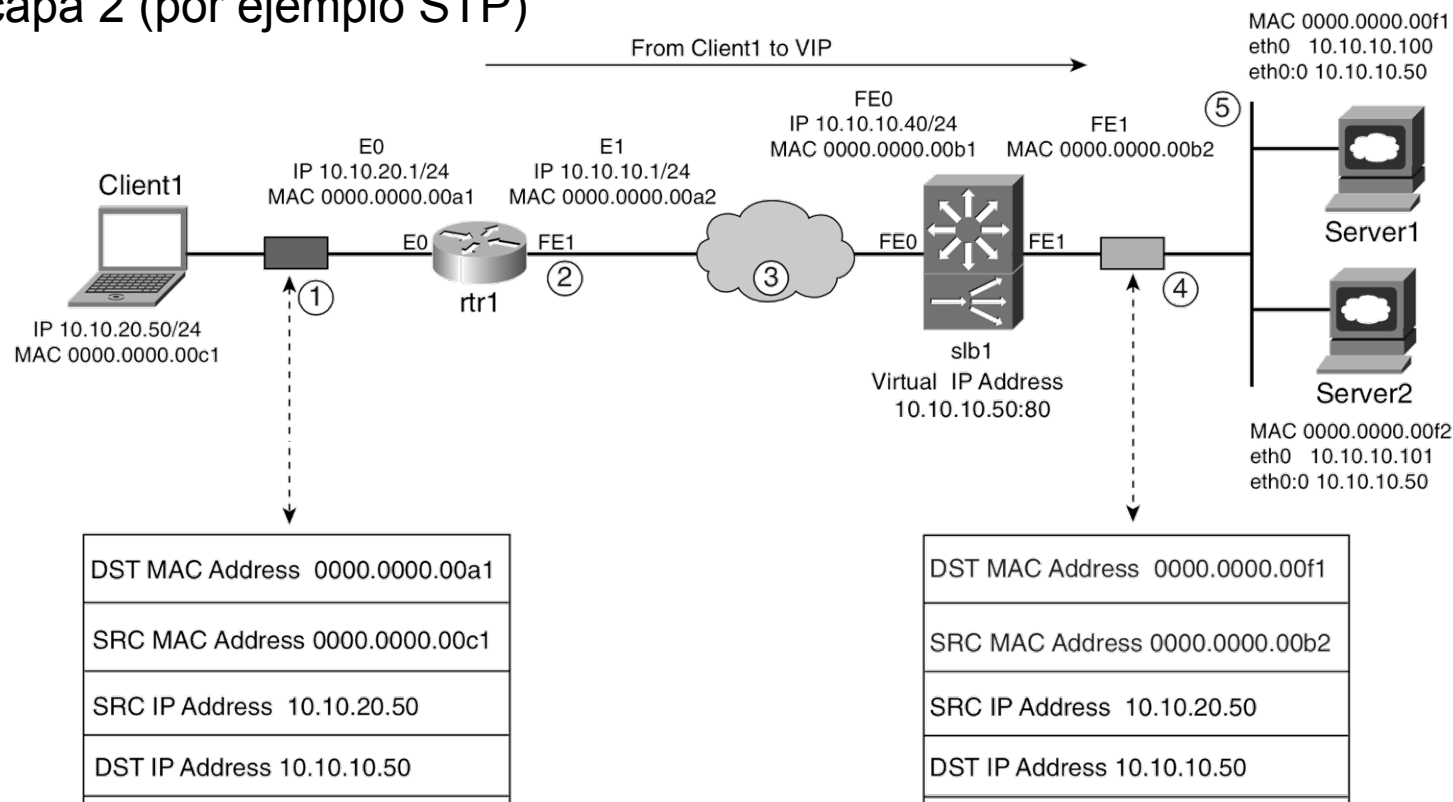
- Normalmente *virtualizan* el servicio ofrecido tomando ellos su dirección IP
- Ocultan las direcciones de los servidores, que normalmente son privadas
- Se puede retirar un servidor para mantenimiento sin detener el servicio
- Se puede añadir nuevos servidores si la capacidad es insuficiente
- Diferentes formas de implementar esta virtualización



Balanceadores: Comportamiento básico

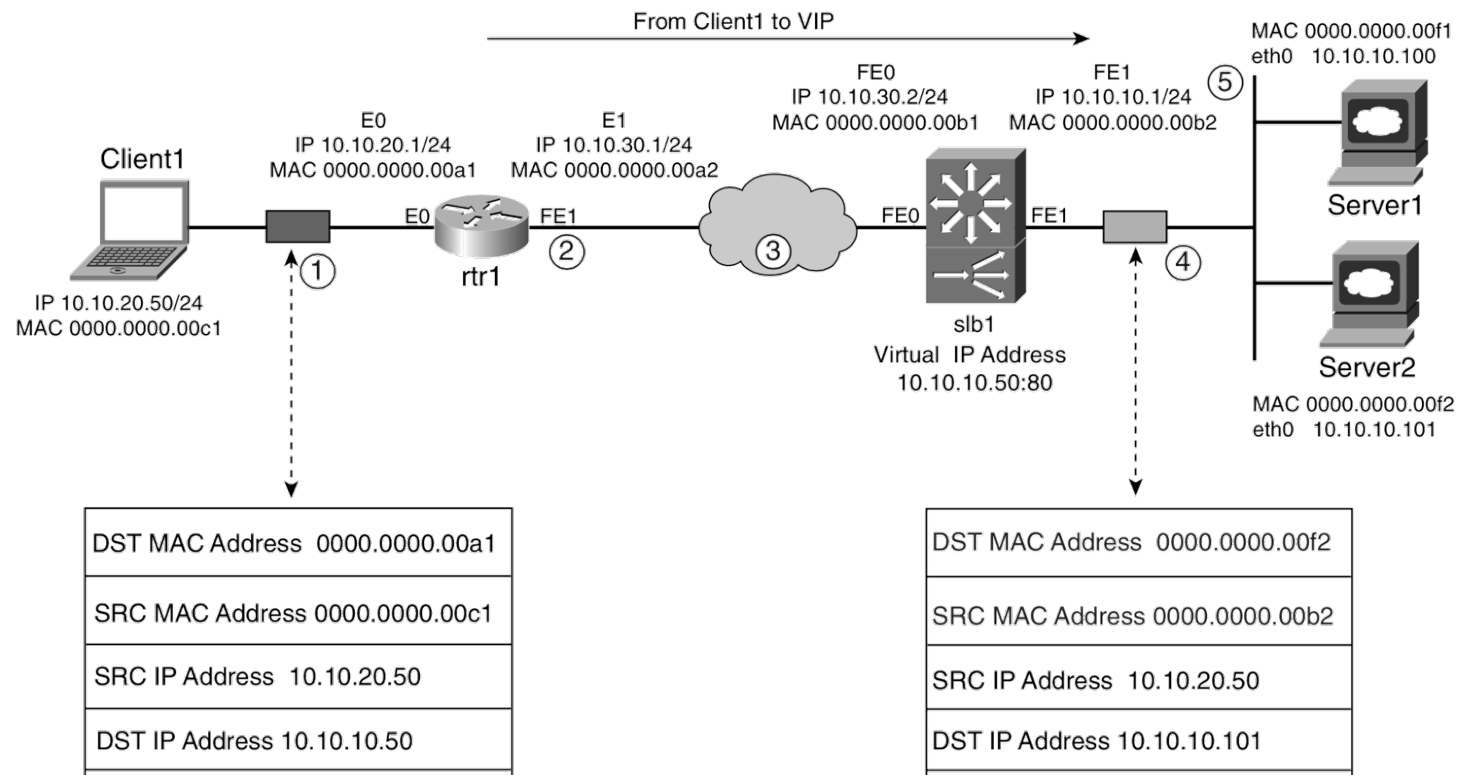
Balanceadores: *Dispatch mode*

- Modifica MAC src por la suya (o no) y dst por la del servidor elegido
- Todos los servidores tienen configurada la VIP como secundaria
- Retorno por router por defecto
- Balanceador y servidores adyacentes en capa 2
- Balanceador no suele tener todas las funcionalidades de un switch capa 2 (por ejemplo STP)



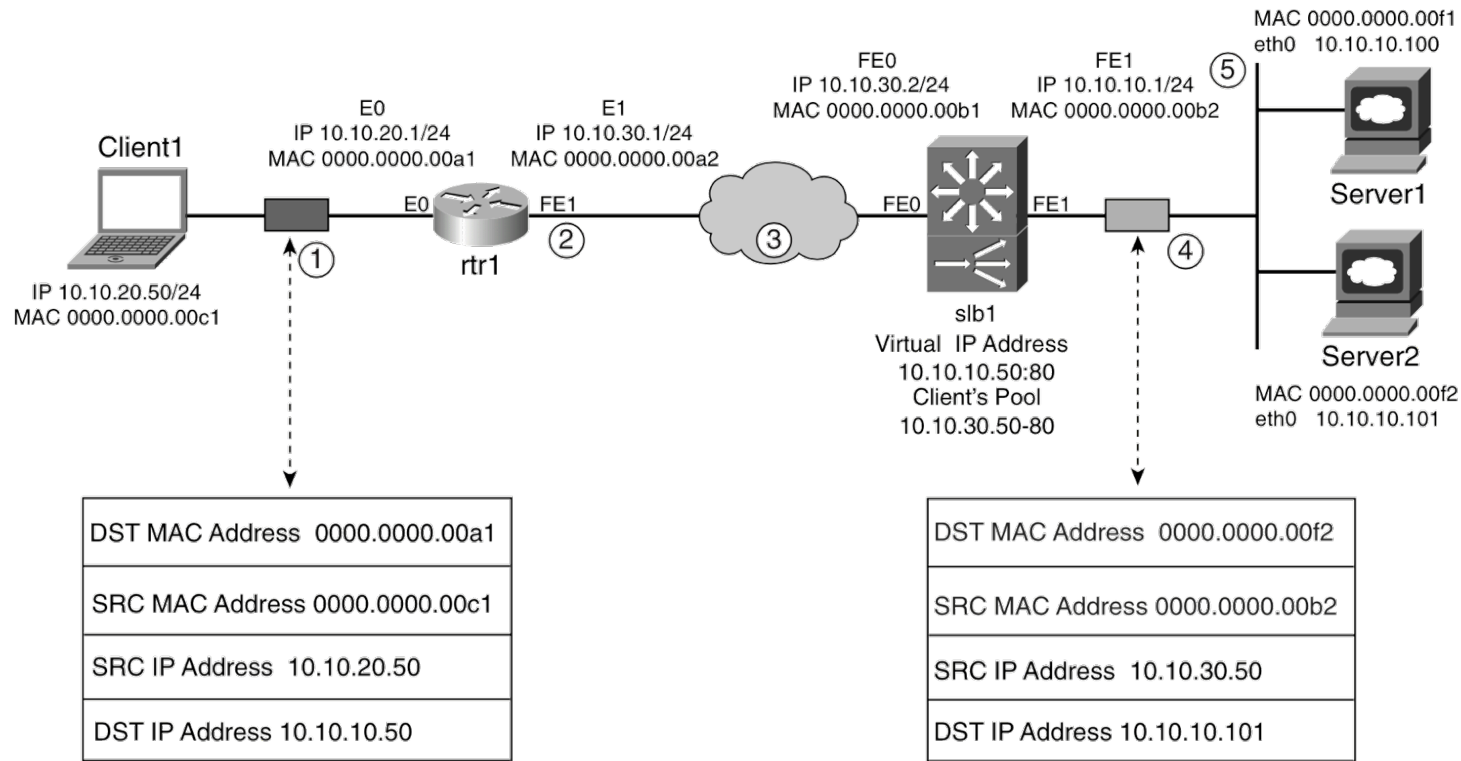
Balanceadores: *Server NAT mode*

- Modifica direcciones MAC y la dirección IP destino
- Los servidores pueden estar en otra subred, a varios saltos
- Los servidores no necesitan tener configurada la VIP
- El tráfico de retorno debe pasar por el balanceador para deshacer el cambio de dirección IP



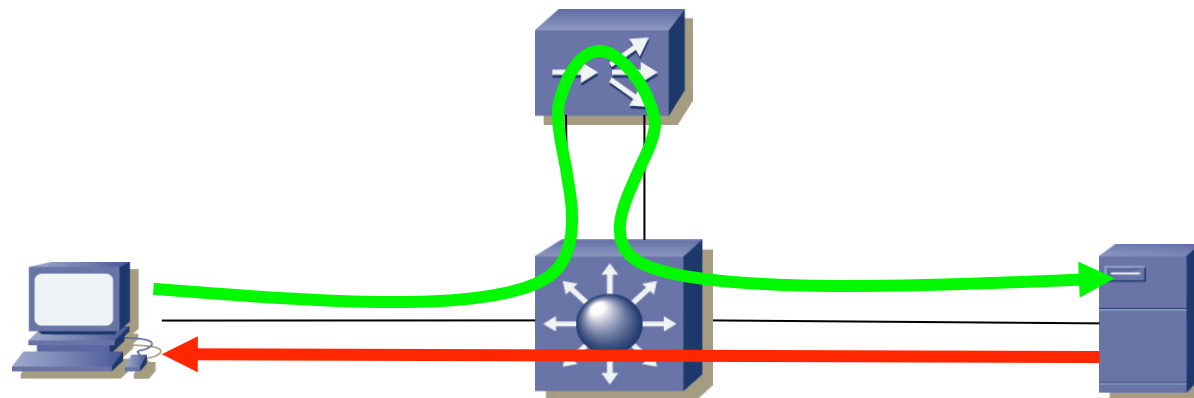
Balanceadores: *Client NAT*

- Se modifica la dirección IP origen
- Aplica a ambos modos anteriores
- Simplifica el conseguir que el tráfico de retorno pase por el balanceador
- Los servidores dejan de conocer la dirección IP del cliente (de cara a hacer estadísticas)



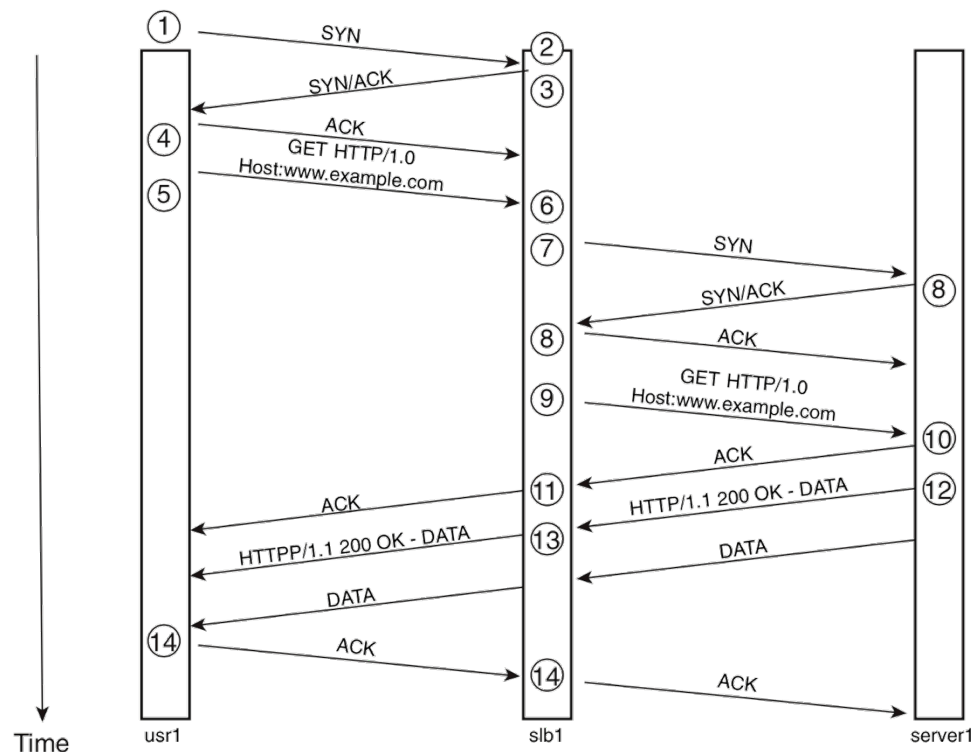
Balanceadores: *Direct Server Return*

- El tráfico de retorno NO pasa por el balanceador
- El balanceador entonces no puede más que reescribir direcciones MAC
- No puede modificar nada en capas 3+, nada que deba deshacerse
- Los servidores deben tener todos configurada la dirección VIP
- No debe caducar el estado de la conexión simplemente por no ver el tráfico en el otro sentido
- Requiere menos trabajo del balanceador y por lo tanto soporta mayores cargas



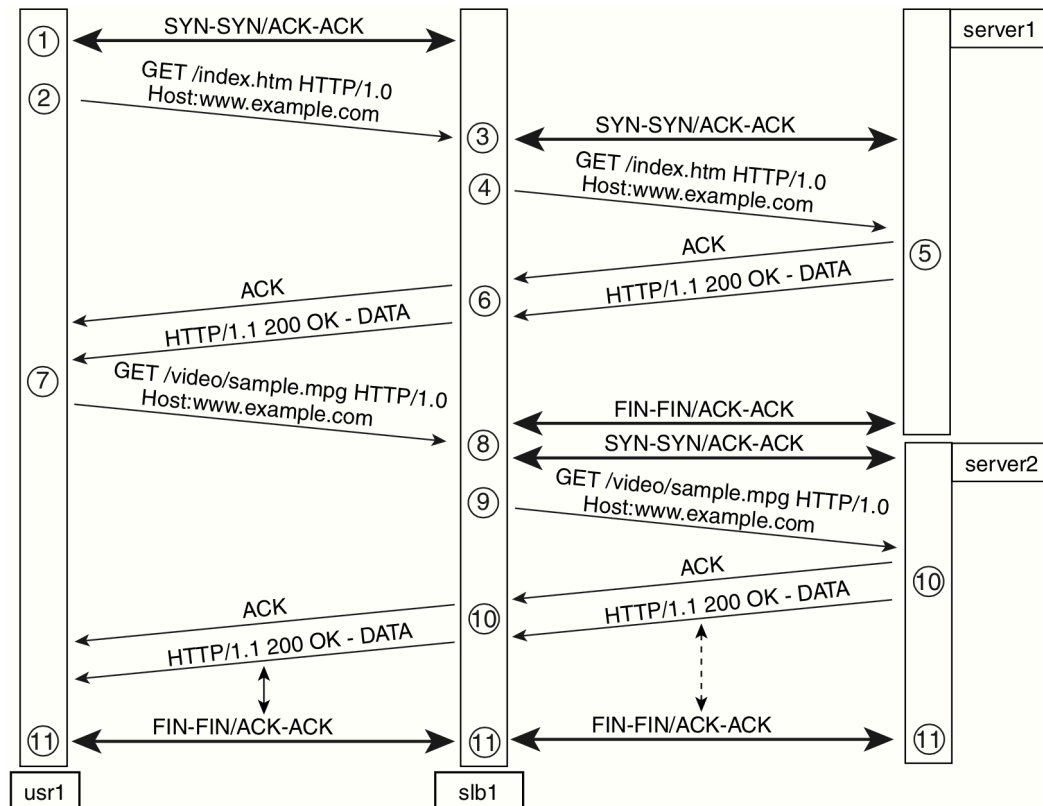
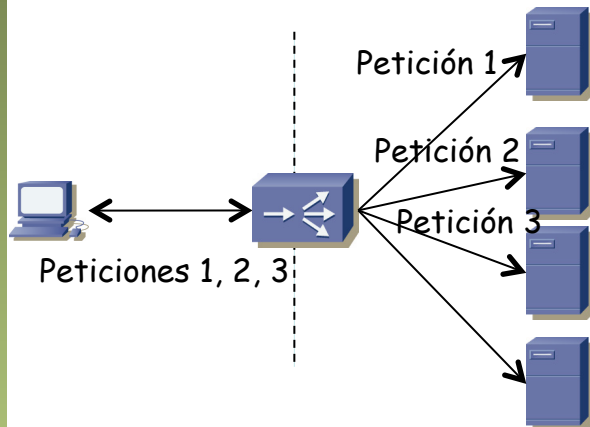
Balanceadores: *Connection Spoofing*

- “TCP termination”, “delayed binding”, “connection splicing”
- En estos casos el balanceador hace de proxy
- Es decir, termina la conexión de cara al cliente y la inicia él de cara al servidor
- Esto le permite elegir el servidor en función de información de capa 5+
- Por ejemplo en HTTP en función del URL



Balanceadores: *Connection Spoofing*

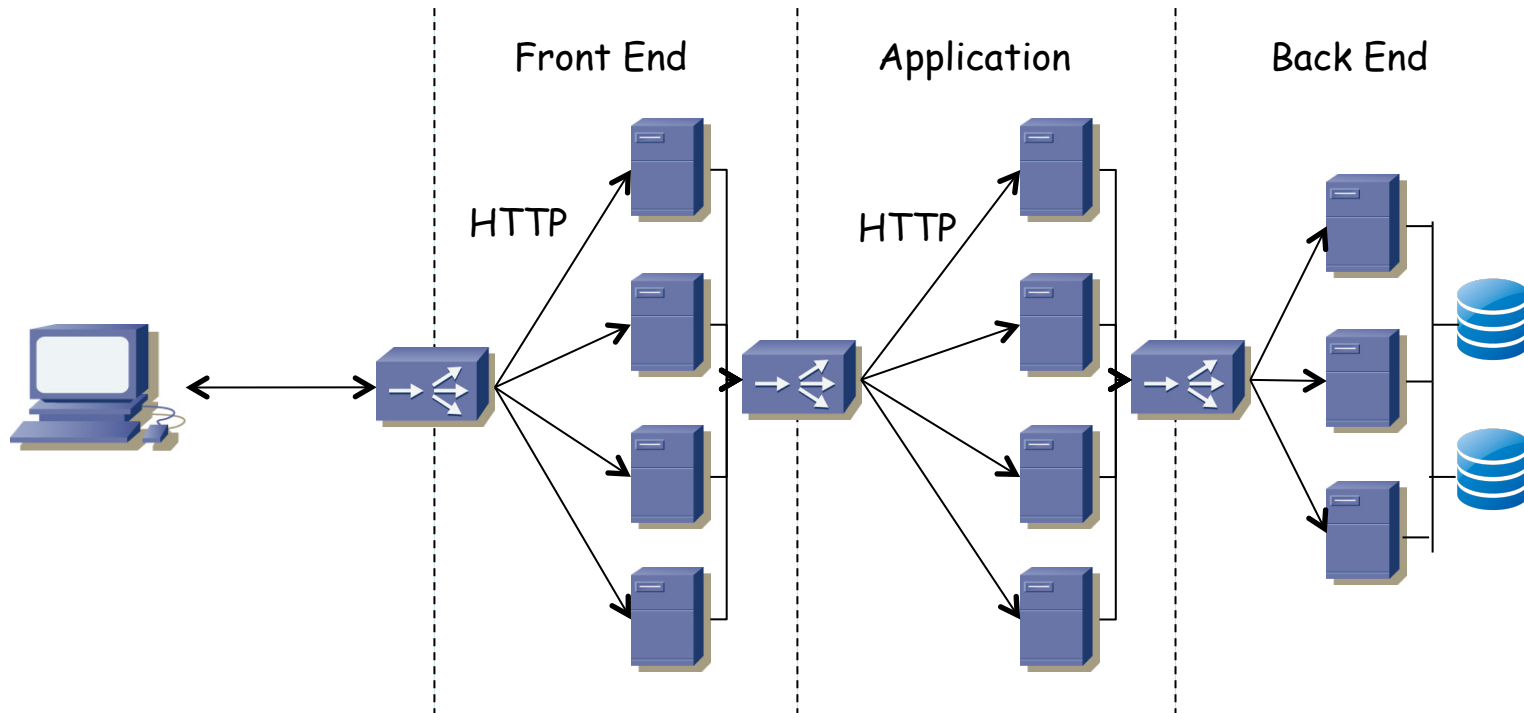
- “*Connection remapping*”: para conexiones persistentes en HTTP
- El balanceador puede repartir la carga de las peticiones entre diferentes servidores (si no requiere mantener la sesión)
- Pueden ser peticiones de recursos diferentes que deban ir a diferentes granjas de servidores aunque se publiquen tras la misma VIP



Balanceadores y HTTP

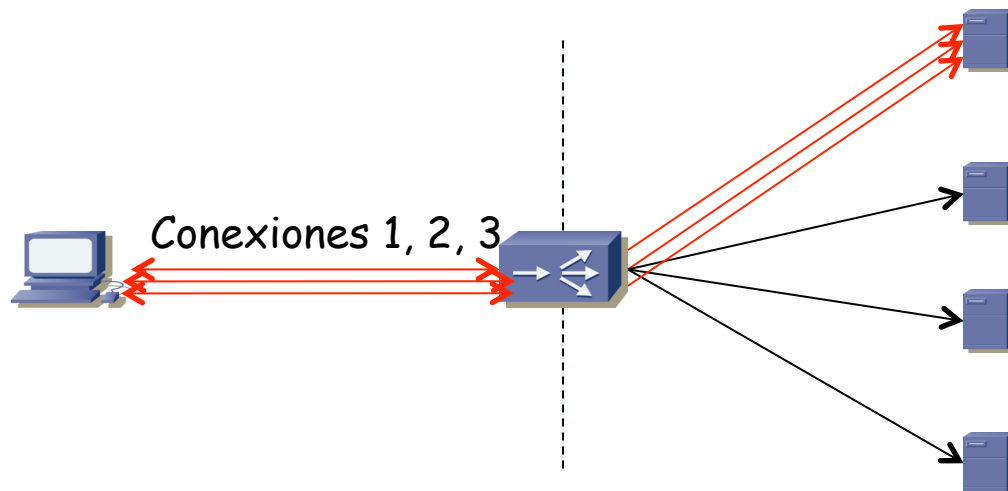
Balanceadores y HTTP

- HTTP es empleado hacia el *front-end*
- También en *Web Services* y por lo tanto en capas de aplicación o del *back-end*



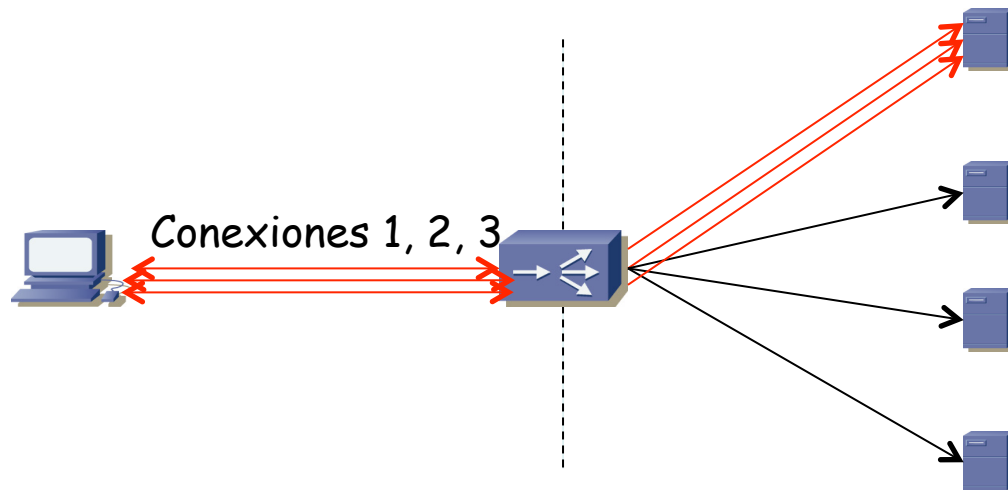
Balanceadores y HTTP

- Con portales web suele hacer falta que todas las conexiones del mismo cliente vayan al mismo servidor
- Esto es así para que se pueda mantener la sesión (por ejemplo el carrito de la compra) sin emplear *clustering* entre servidores
- Con web services no suele haber necesidad
- “*Session persistance*”, “*stickiness*”, “*sticky connections*”
- Hay varias técnicas para conseguirlo, que se centran en cómo reconocer al cliente (...)



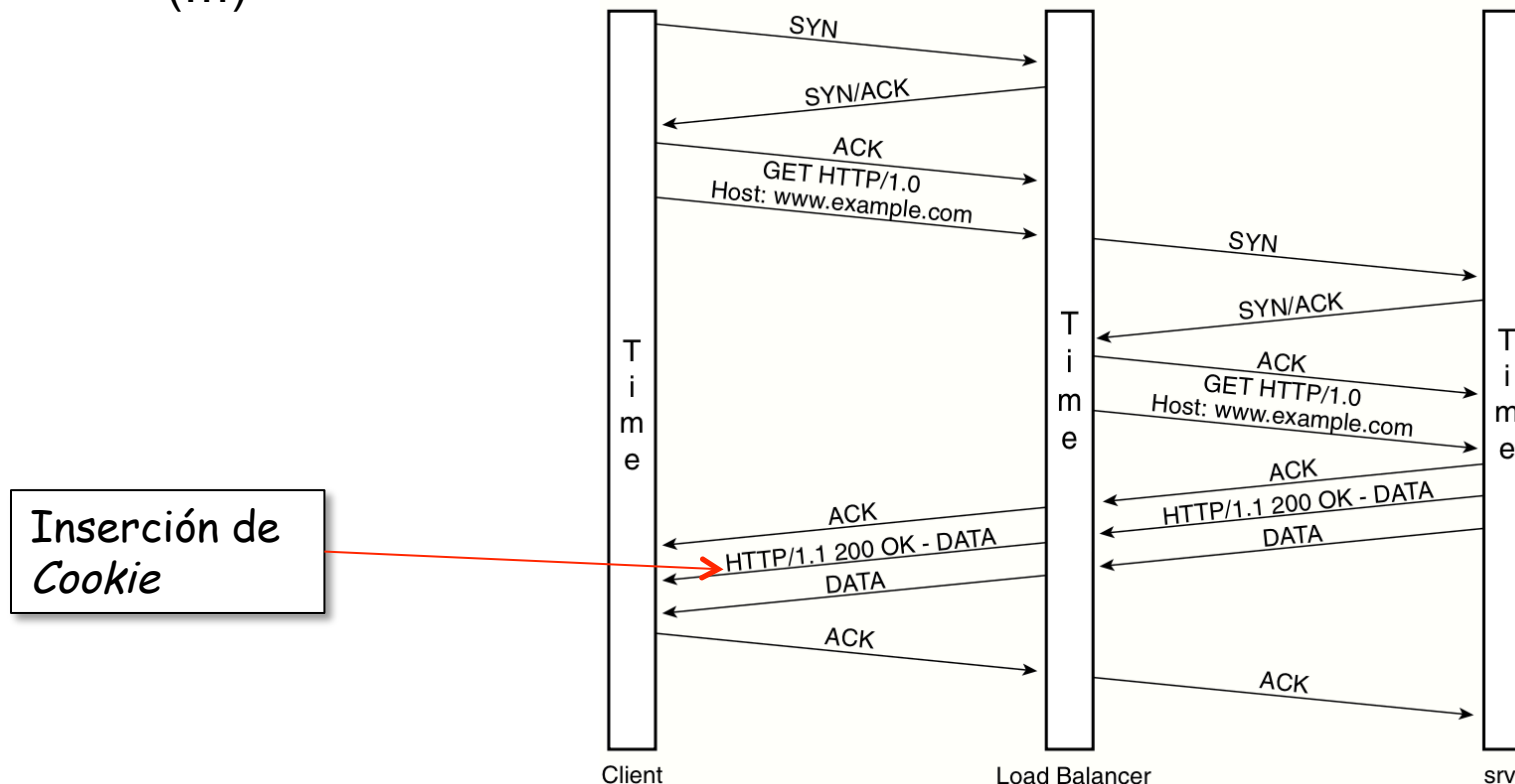
Session persistence

- *Source IP sticky*: basarse en la dirección IP origen
 - Es factible en entornos donde se tienen controlados a los clientes (una Intranet)
 - Es problemático si los clientes son cualquiera en Internet porque
 - Podría el cliente estar empleando una granja de proxies que hicieran que varias conexiones vinieran con diferentes IP origen
 - Pero al menos funciona aunque se emplee HTTPS desde el cliente hasta el servidor



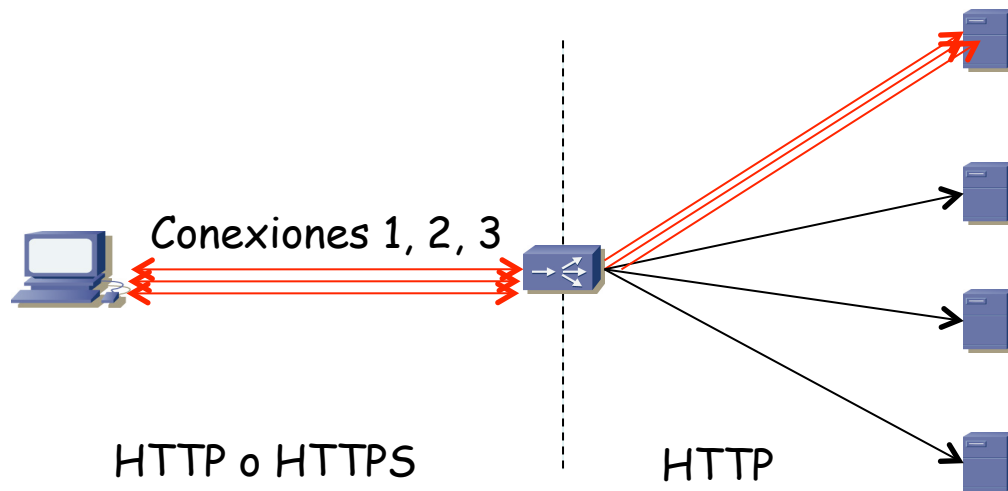
Session persistence

- *Cookie sticky*: inserción de *cookies*
 - El balanceador inserta una *cookie* en la respuesta (*cookie insert*)
 - O la puede insertar el propio servidor (*cookie passive*)
 - Esta cookie le permite al balanceador reconocer al cliente en la siguiente petición para mandarla al mismo servidor
 - (...)



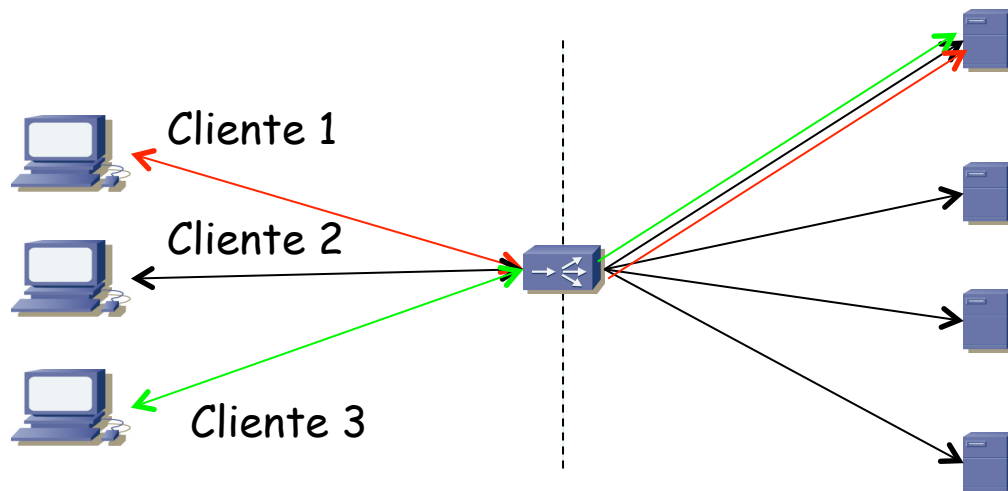
Session persistence

- *Cookie sticky*: inserción de *cookies*
 - La comunicación con el servidor debe ser sin encriptar para poder ver la cookie (si la inserta él) o poder insertarla
 - O el balanceador debe disponer de la clave privada para desencriptar y re-encryptar el flujo SSL del servidor
 - Cuando emplean información de capas 5+ se les suele llamar *content switches*



Balanceadores y HTTP

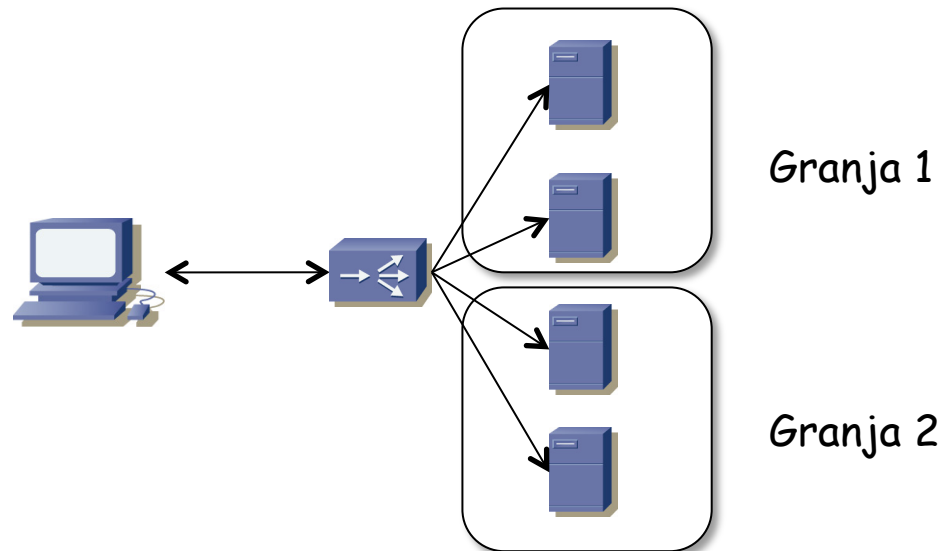
- También puede hacer coalescencia de peticiones de diferentes clientes
- Por ejemplo para aprovechar conexiones persistentes HTTP 1.1 con el servidor y evitar fases de *slow start*
- O para reducir el número de conexiones que mantiene el balanceador con los servidores o que mantienen estos últimos



Balanceadores y reparto de carga

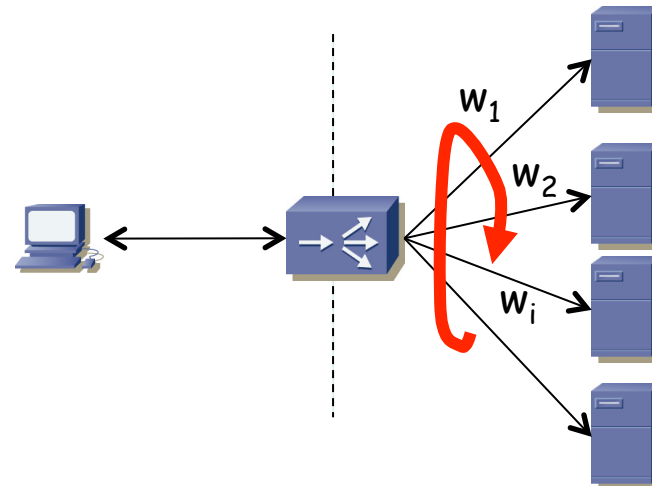
Pasos

1. Seleccionar la granja de servidores
 - Pueden estar en distintos CPDs
 - Suele ser en función de parámetros de la conexión o de capa 5+
 - Por ejemplo puede haber diferentes granjas según el subconjunto de URLs
2. Seleccionar el servidor
 - Algoritmos de balanceo de carga



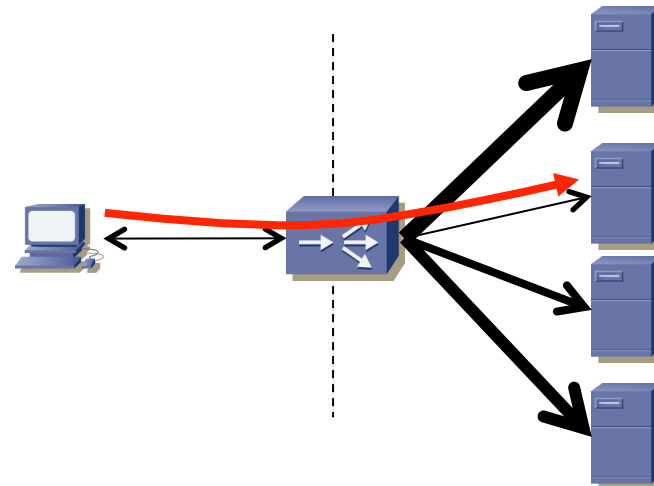
(Weighted) Round Robin

- *Round robin*
 - Una lista de servidores disponibles
 - Cada nueva petición se entrega al siguiente de la lista
 - De forma circular
- *Weighted round robin*
 - Un peso por servidor controla cuántas peticiones recibe ese servidor en un turno
 - Permitiría equilibrar la carga de trabajo cuando los servidores son de diferentes características técnicas



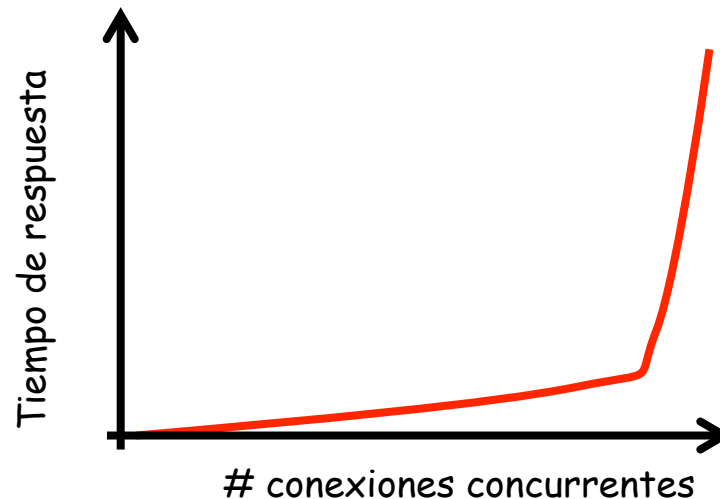
(Weighted) Least Connections

- *Least connections*
 - Se envía la nueva conexión al servidor con el cual el balanceador tiene menos conexiones
- *Weighted least connections*
 - Se ordena a los servidores en base a “nº conexiones”/weight
 - Se selecciona al de menor métrica
 - De nuevo para poder diferenciar servidores con distintas capacidades



Fastest

- Se hace una estimación de lo “rápidos” que son los servidores en contestar al SYN
- Es decir, una media de los tiempos de SYN a SYN+ACK
- Normalmente al aumentar la carga del servidor aumenta ese tiempo
- Suele haber un punto a partir del cual empeora rápidamente
- Puede que el balanceador no reaccione lo suficientemente rápido



Otros esquemas

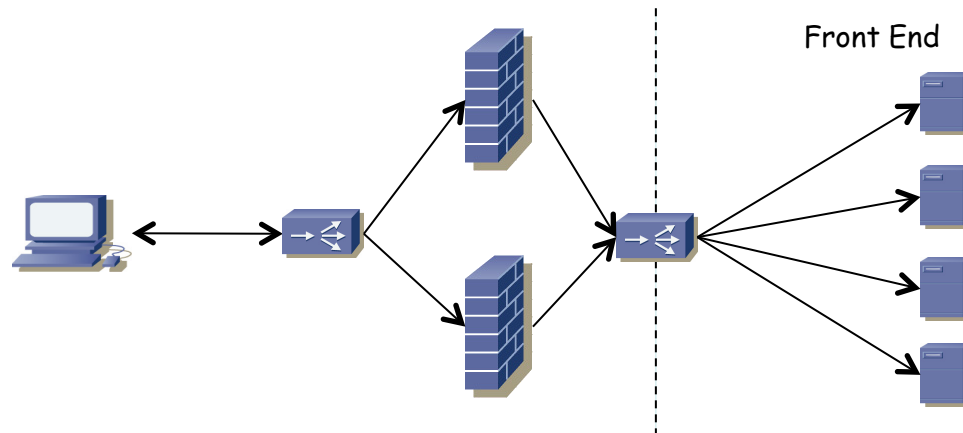
- En función de la dirección IP del cliente
- En base a un hash en función de las direcciones IP origen y destino (cuando hay varias VIPs)
- En función del URL (o del tipo de recurso: imagen, html, etc)
- En base a un hash del URL



Otros servicios

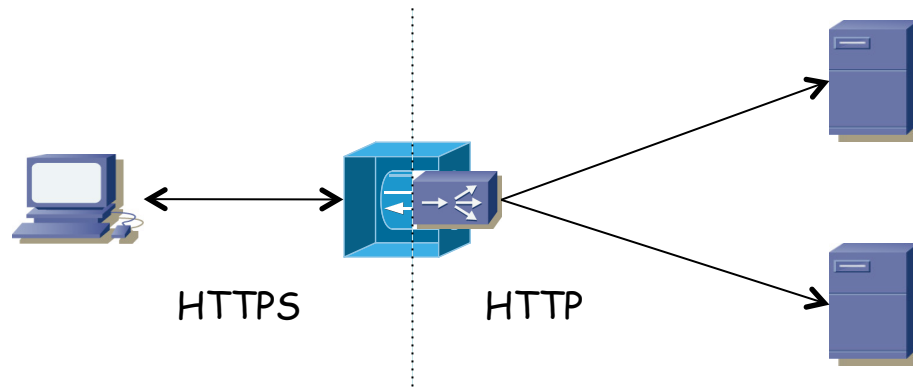
Firewalls e IDS

- Seguridad, seguridad, seguridad
- Reglas de filtrado para permitir el acceso solo a las direcciones IP y puertos de los servicios
- Inspección de contenido
- Pueden estar antes o después del balanceador
- Si no vale con uno se pueden poner varios balanceados (aumenta la complejidad)
- Ese balanceador podría ser el mismo que hacia los servidores (varias direcciones IP virtuales o instancias virtuales)



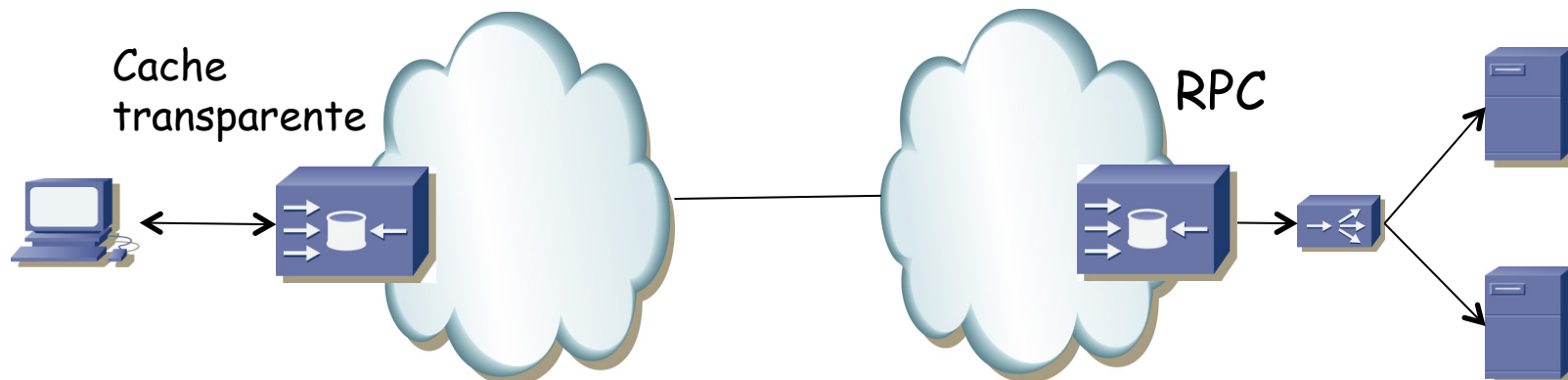
SSL offloading

- Portales web seguros
- También otros servicios sobre un túnel SSL
- SSL tiene un coste computacional considerable
- Este equipo termina la sesión SSL con el usuario e inicia una conexión sin SSL con el servidor
- El equipo puede disponer de hardware especializado para SSL
- También podríamos poner varios y balancearlos
- Es común que el balanceador integre esta funcionalidad



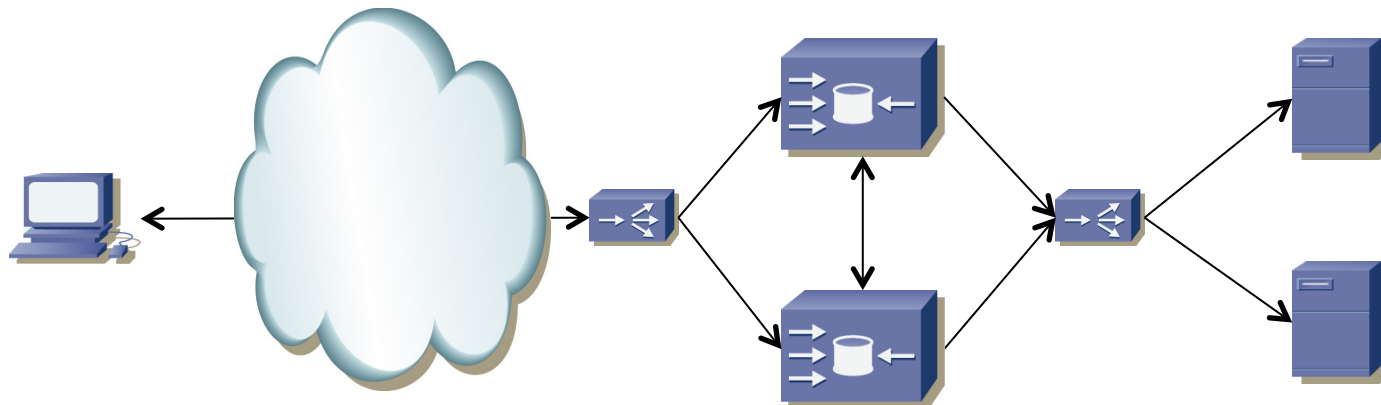
Cache

- Puede ser cercana a los servidores, a los clientes o a ambos
- Cercanas al servidor
 - Se habla de “*reverse proxy cache*” (RPC)
 - Reduce carga sobre los servidores
- Cercanas al cliente
 - Se habla de “*transparent caching*”
 - Reducen carga sobre el enlace a Internet
 - Reducen tiempos de respuesta por cercanía (menor RTT)



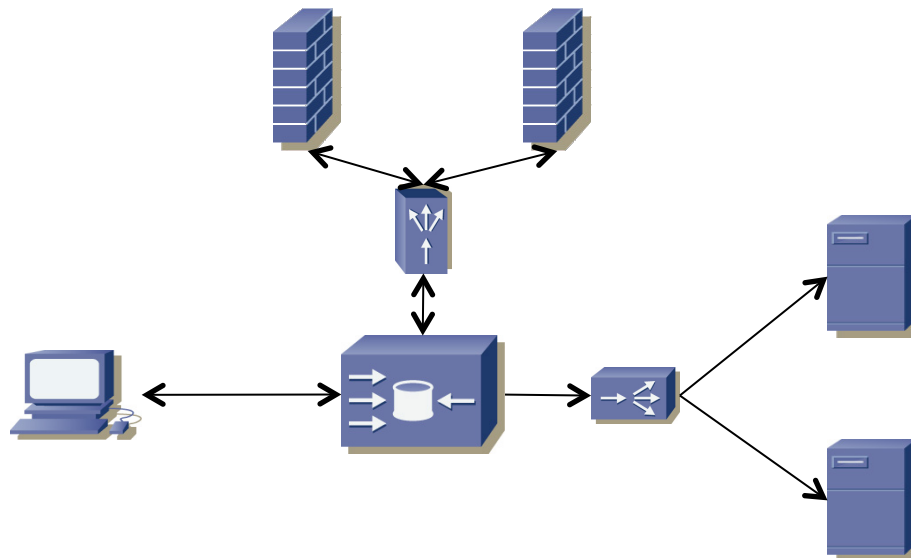
Cache

- La cache podría implementarse con varias caches balanceadas
 - Aumenta la capacidad (CPU) de la cache
 - Busca maximizar el *cache hit ratio* y así reducir peticiones a servidores
 - Para ello el balanceador debería reenviar la petición a la cache con mayor probabilidad de contenerlo (en función del FQDN)
 - O las caches deben sincronizarse (*clustering*), pues si no acabarán haciendo peticiones repetidas



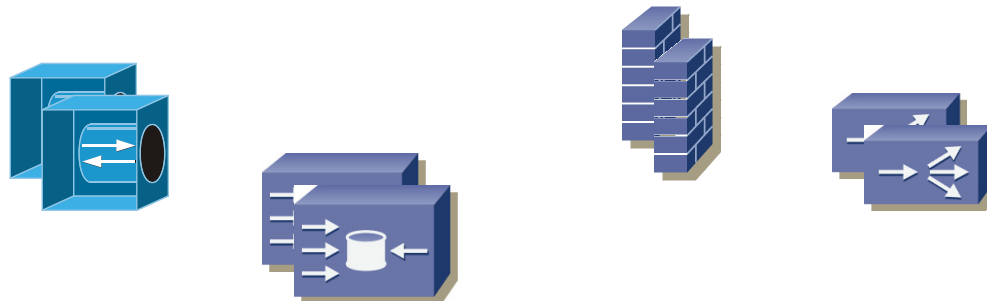
Cache

- Puede redirigir parte de la petición a un antivirus o filtro de contenido
- Se encargaría de verificar que se puede hacer esa petición o que el documento obtenido no es peligroso
- Protocolos específicos para pasar la petición o respuesta: ICAP = *Internet Content Adaptation Protocol* (RFC 3507)
- O a varios con balanceo de carga
- El balanceador puede ser el mismo equipo



Servicios y redundancia

- Todos estos servicios se pueden dar desde equipos independientes
- Si no queremos un punto único de fallo debemos tenerlos replicados
- Según el tipo de servicio deberán coordinarse entre ellos para mantener el estado ante un fallo
- Por ejemplo un NAT para conocer las sesiones de mapeo que estaban establecidas



Servicios y redundancia

- Todos estos servicios se pueden dar desde equipos independientes
- Si no queremos un punto único de fallo debemos tenerlos replicados
- Según el tipo de servicio deberán coordinarse entre ellos para mantener el estado ante un fallo
- Por ejemplo un NAT para conocer las sesiones de mapeo que estaban establecidas
- También pueden ser módulos en un conmutador

