IPv6 para VMs en Proxmox sobre OVH bare-metal (NDP proxy)

- **Fecha:** 2026-05-23
- **Autor:** Abdelkarim Mateos
- **Estado:** vigente
- **Aplicable a:** servidores OVH bare-metal con dos NICs separadas (RISE / Advance / SYS) y Proxmox VE 8/9 como hypervisor KVM
- **No aplica:** OVH HG (NIC única + vRack en VLAN tagged), OVH Public Cloud, Hetzner Cloud o cualquier proveedor donde el switch del segmento público no haga filtrado por MAC

> **Nota sobre las direcciones de los ejemplos.** Todas las IPv4, IPv6 y MAC que aparecen en este documento son **valores de documentación reservados por la IETF**: bloque `203.0.113.0/24` (RFC 5737, TEST-NET-3) para IPv4 y prefijo `2001:db8::/32` (RFC 3849) para IPv6. Sustituir por los valores reales del servidor al aplicar. He preservado deliberadamente la **topología característica de OVH** — gateway IPv6 con muchos hextets `ff` y fuera del `/64` ruteado, requiriendo `onlink` — para que los ejemplos reflejen el comportamiento real. Publicar IPs reales en documentación pública es un señuelo gratuito para scanners — evítalo siempre.

Escenario

Pongo en marcha un OVH RISE-L (Ryzen 9, 128 GB RAM, 2× Intel I210) como host Proxmox VE 9 para alojar máquinas virtuales KVM. La intención: que las VMs tengan IPv6 público aprovechando el `/64` que OVH enruta por defecto hacia el MAC del servidor.

El primer intento (asignar a la VM una dirección del `/64` con el gateway OVH `onlink`, igual que se hace en Hetzner Cloud) **no funciona** — el NDP de la VM al gateway se queda en `INCOMPLETE`. Este documento explica por qué y cómo se resuelve con NDP proxy (`ndppd`) más un bridge interno.

Topología hardware OVH RISE

Las gamas **RISE / Advance / SYS** entregan **dos NICs separadas**, no agrupadas en LACP, sin bonding. Una va al switch público de OVH, la otra al vRack (red privada / IPs adicionales enrutadas).

Modelo	Topología NICs
RISE / Advance / SYS	2× NICs separadas — una pública, otra vRack
HG (High Grade / Game)	NIC única, vRack en VLAN tagged sobre la misma interfaz
Eco	Igual que RISE en topología, con limitaciones de garantía vRack

En este documento la NIC pública es `enp6s0` (mapeada al bridge `vmbr0`) y la NIC del vRack es `enp7s0` (mapeada al bridge `vmbr1`). Cada modelo usa nombres distintos según firmware/kernel — comprobar con `lspci -nn | grep Ether` y `ip -br link`.

Recursos IP del ejemplo (documentación)

- `vmbr0` **(pública):** IPv4 del bloque que OVH asigna al servidor (`203.0.113.10/24`) y un `/64` **IPv6 enrutado al MAC del servidor** (`2001:db8:1:1::/64`, gateway `2001:db8:1:ffff:ff:ff:ff:ff`).
- `vmbr1` **(vRack):** sin IP en el host, bridge limpio. Las VMs cogen IPs de los bloques que el cliente haya pedido enrutar al vRack.

Por qué el approach naive no funciona

Asignar a una VM una IPv6 del `/64` con el gateway OVH `onlink` es lo que recomienda la documentación oficial OVH para el lado guest. Pero falla:

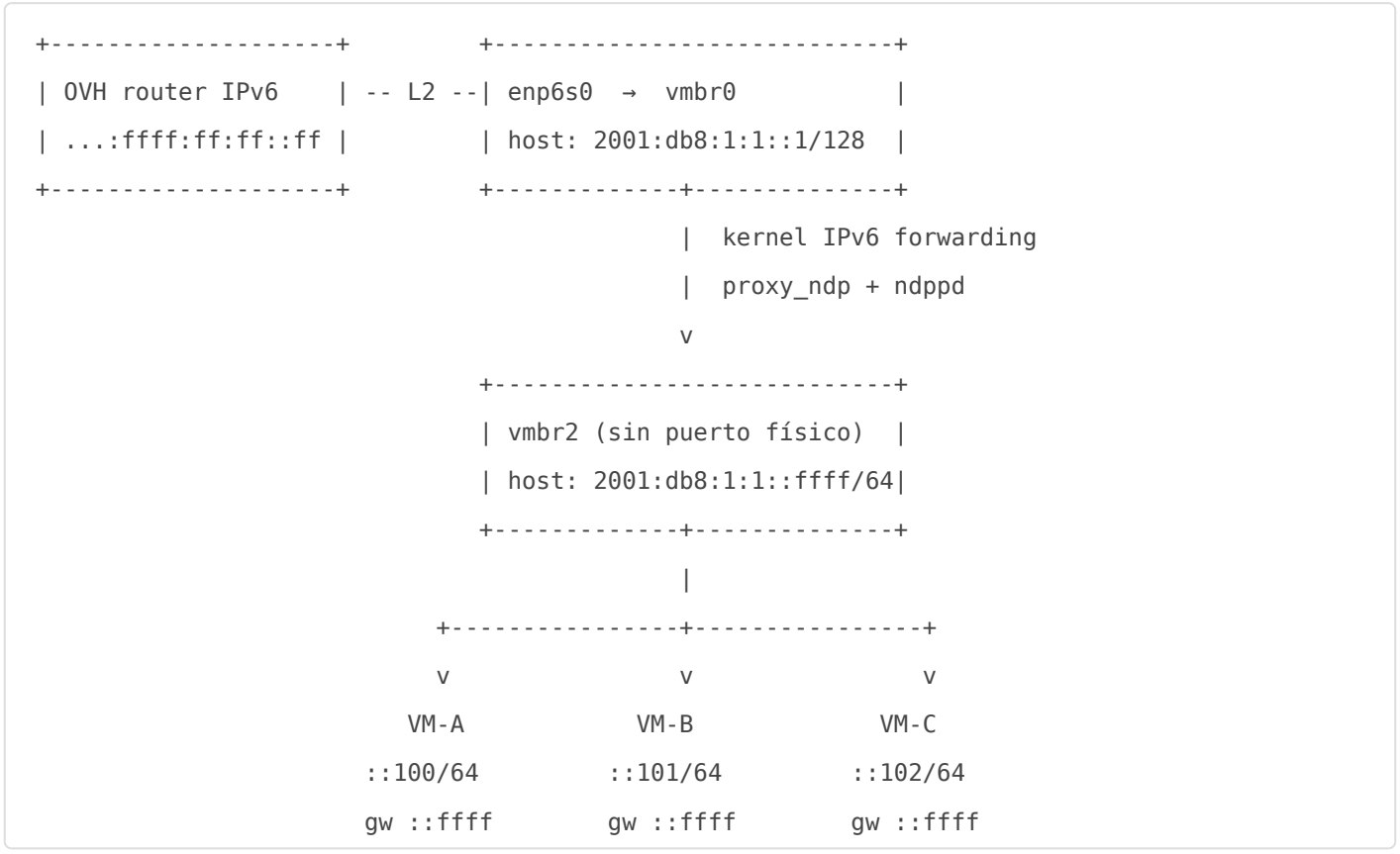
```
admin@vm:~$ ping6 2001:db8:1:ffff:ff:ff:ff:ff
From 2001:db8:1:1::100 icmp_seq=1 Destination unreachable: Address unreachable


admin@vm:~$ ip -6 neigh show
2001:db8:1:ffff:ff:ff:ff:ff dev eth1 INCOMPLETE
```

La VM envía una *Neighbor Solicitation* al gateway y **nunca recibe Neighbor Advertisement** de vuelta. Causa: el switch del segmento público de OVH hace **filtrado por MAC**. Solo deja entrar tramas cuyo MAC destino sea el del servidor — la respuesta del gateway, dirigida al MAC de la VM, se descarta en el switch.

En el vRack ese filtrado no está activo (es un L2 abierto, precisamente para esto). Por eso el IPv4 funciona "directo" en el vRack pero el IPv6 del `/64` público no.

Arquitectura NDP proxy

La solución estándar es que **el host responda NDP por las VMs**. Ante el switch OVH solo existe un MAC (el del host); el routing IPv6 a cada VM se hace internamente en el kernel del host:



Tres piezas:

1. **Host con `/128` en `vmbr0`** — no se queda con el `/64` entero, solo con una IP propia (típicamente `::1/128` o `::ffff/128`).
2. **Bridge interno `vmbr2`** sin puerto físico, con el `/64` y una IP que actúa de gateway para las VMs (en este ejemplo `::ffff`).
3. **`ndppd` escuchando en `vmbr0`** — responde a las NDP Solicitations del gateway OVH con el MAC del host para cualquier dirección dentro del `/64`. El kernel entonces enruta el tráfico hacia `vmbr2` por la tabla v6, y de ahí a la VM correspondiente.

Sysctl obligatorios:

- `net.ipv6.conf.all.forwarding = 1` — sin esto el host descarta paquetes IPv6 destinados a otras IPs.

- `net.ipv6.conf.all.proxy_ndp = 1` — habilita la lógica de NDP proxy del kernel. `ndppd` lo activa al arrancar pero conviene persistirlo por sysctl.

Configuración paso a paso

1. `/etc/network/interfaces` del host

Stanza relevante (Proxmox autogestiona la primera parte; estas líneas se añaden al final):

```
# IPv6 NDP-proxy approach
iface vmbr0 inet6 static
    address 2001:db8:1:1::1/128
    gateway 2001:db8:1:ffff:ff:ff:ff:ff

auto vmbr2
iface vmbr2 inet6 static
    address 2001:db8:1:1::ffff/64
    bridge-ports none
    bridge-stp off
    bridge-fd 0
```

Aplicar con `ifreload -a` — no hace falta reiniciar.

2. Sysctl persistente

`/etc/sysctl.d/99-ipv6-ndp-proxy.conf`:

```
net.ipv6.conf.all.forwarding = 1
net.ipv6.conf.all.proxy_ndp = 1
```

Aplicar con `sysctl --system`.

3. Instalar y configurar `ndppd`

```
apt-get install -y ndppd
```

`/etc/ndppd.conf`:

```
route-ttl 30000

proxy vmbr0 {
   router no
   timeout 500
   ttl 30000
   rule 2001:db8:1:1::/64 {
      auto
   }
}
```

Trampa de versión: el ejemplo distribuido en `/usr/share/doc/ndppd/ndppd.conf-dist` menciona el keyword `static` como "NEW", pero en la versión que trae Debian 13 Trixie (`ndppd 0.2.4`) **`static` falla con `Failed to load configuration file`** sin más detalle. Usar `auto` — funciona y resuelve dinámicamente vía `/proc/net/ipv6_route`.

Habilitar el servicio:

```
systemctl enable --now ndppd
systemctl is-active ndppd       # debe responder 'active'
journalctl -u ndppd -n 20       # 'Failed to load' = error de sintaxis
```

4. Configurar las VMs

Por cloud-init en Proxmox:

```
qm set <VMID> --net1 virtio,bridge=vmbr2
qm set <VMID> --ipconfig1 "ip6=2001:db8:1:1::100/64,gw6=2001:db8:1:1::ffff"
```

Importante: el **gateway de la VM es la IP del host en `vmbr2`** (`::ffff`), NO el gateway OVH. La VM rutea hacia el host, el host hace el forwarding hacia OVH.

Para Debian/Ubuntu sin cloud-init, en `/etc/network/interfaces` de la VM:

```
iface ens19 inet6 static
    address 2001:db8:1:1::100/64
    gateway 2001:db8:1:1::ffff
```

5. CSF (si se usa)

Si el host tiene CSF con `IPV6 = "1"`, CSF deja la cadena `FORWARD` de `ip6tables` en `policy DROP`. Esto **rompe el forwarding** entre `vmbr2` y `vmbr0` — las VMs reciben IPv6 pero no pueden enviar.

Añadir un hook persistente en `/usr/local/include/csf/post.d/10-ipv6-forward.sh`:

```
#!/bin/sh
ip6tables -A FORWARD -i vmbr2 -o vmbr0 -j ACCEPT
ip6tables -A FORWARD -i vmbr0 -o vmbr2 -j ACCEPT
```

`chmod +x` y `csf -r`. CSF carga `post.d/*.sh` automáticamente tras cada restart.

Verificación

Desde dentro de una VM con la configuración aplicada:

```
admin@vm:~$ ip -6 addr show eth1 | grep inet6
    inet6 2001:db8:1:1::100/64 scope global

admin@vm:~$ ping6 -c2 2606:4700:4700::1111
PING 2606:4700:4700::1111 56 data bytes
64 bytes from 2606:4700:4700::1111: icmp_seq=1 ttl=58 time=5.13 ms
64 bytes from 2606:4700:4700::1111: icmp_seq=2 ttl=58 time=5.14 ms

admin@vm:~$ ip -6 route show
2001:db8:1:1::/64 dev eth1 proto kernel
default via 2001:db8:1:1::ffff dev eth1 proto static
```

Desde el host:

```
# El gateway OVH debe aparecer REACHABLE, no INCOMPLETE
root@host:~$ ip -6 neigh show dev vmbr0 | grep ffff:ff
2001:db8:1:ffff:ff:ff:ff:ff lladdr 02:00:5e:00:00:01 router REACHABLE

# ndppd activo
root@host:~$ pidof ndppd
12345

# La tabla NDP proxy estática puede estar vacía — ndppd responde dinámicamente
root@host:~$ ip -6 neigh show proxy
```

```
(vacío)
```

Trampas conocidas

- **Happy Eyeballs ignora `gai.conf`.** Si necesitas que el host hable solo IPv4 (por ejemplo para que `acme.sh` use un token Cloudflare con allowlist solo IPv4), el típico `precedence ::ffff:0:0/96 100` en `/etc/gai.conf` no basta: `curl` con Happy Eyeballs (RFC 8305) lanza la conexión IPv6 con 200 ms de ventaja y siempre gana contra anycast. Para forzar IPv4 hay que quitar IPv6 del host o usar `curl -4` por comando.
- **`ndppd static` no funciona en 0.2.4 (Debian 13).** Usar `auto`.
- **CSF y `FORWARD policy DROP`** — necesitan hook en `post.d`. Sin él, las VMs reciben IPv6 pero no pueden enviar (FORWARD bloquea ambos sentidos para flujos nuevos).
- **`qm shutdown` falla sin `qemu-guest-agent`** en la VM. Las imágenes Ubuntu cloud no lo traen instalado por defecto — instalar `qemu-guest-agent` dentro de la VM si quieres shutdown limpio.
- **MAC regenerado al cambiar de bridge.** Si haces `qm set <VMID> --net1 virtio,bridge=vmbrX` sin especificar `=<MAC>` previo, Proxmox regenera el MAC. Cloud-init reescribe netplan en consecuencia.
- **Una entrada NDP proxy por VM vs `ndppd` para todo el `/64`.** Si tienes pocas VMs estables, puedes prescindir de `ndppd` y poner `ip -6 neigh add proxy <ip> dev vmbr0` (como `post-up` en la stanza de `vmbr2`). Para entornos dinámicos `ndppd` con regla `auto` sobre el `/64` es más cómodo.

Limitaciones / cuándo NO usar este approach

- **OVH HG (NIC única + vRack VLAN):** el bridge debe configurarse sobre la subinterfaz tagged, y la dinámica puede diferir. No verificado aquí.
- **vRack IPv6** (servicio separado de OVH que enruta un `/56` específicamente al vRack): si lo tienes contratado es preferible — las VMs pueden estar directamente en `vmbr1` con un `/64` propio y sin NDP proxy. Este documento NO cubre ese escenario.
- **Más de un host con el mismo `/64`:** el `/64` IPv6 OVH va enrutado a UN solo MAC. Si quieres HA con failover de IPv6 entre varios servidores necesitas otro patrón (BGP, ULA + NAT66, o IP failover OVH con reasignación manual).

Referencias

- OVH — Configure an IPv6 on a VM — guía oficial OVH (cubre solo el lado guest, no menciona NDP proxy).
- mathieu-gilloots.fr — Installation Proxmox 8 avec IPv6 OVH — referencia detallada del approach NDP proxy con `npd6`.
- greenhoster.fr — IPv6 dans un vRack OVHcloud — variante con bloque IPv6 vRack `/56`.
- RFC 5737 y RFC 3849 — rangos reservados para documentación (IPv4 e IPv6).
- `man ndppd.conf` y `/usr/share/doc/ndppd/ndppd.conf-dist`.

Documento generado: 2026-05-23 — Abdelkarim Mateos / AichaDigital

---

Revision #2
Created 2026-05-23 05:19:08 UTC by Abkrim
Updated 2026-05-23 05:38:54 UTC by Abkrim