

Unknown Title



You said

```
explica: root@R1-ubuntu:~# ip r
default via 10.2.7.1 dev eth207 proto static
10.2.7.0/24 dev eth207 proto kernel scope link src 10.2.7.3
172.18.10.0/24 dev eth217 proto kernel scope link src 172.18.10.1
172.18.20.0/24 dev eth227 proto kernel scope link src 172.18.20.1
172.18.30.0/24 dev eth237 proto kernel scope link src 172.18.30.1
```



Gemini said

Esta es la **tabla de enrutamiento IP** de tu máquina Linux. Básicamente, es el "mapa" que usa el sistema operativo para saber por qué puerta (interfaz) debe enviar los paquetes de datos para que lleguen a su destino.

Aquí tienes el desglose línea por línea:

1. La Ruta por Defecto (Gateway)

```
default via 10.2.7.1 dev eth207 proto static
```

- **Significado:** Es la salida al resto del mundo (Internet o redes externas).
 - **Interpretación:** Si el destino de un paquete no coincide con ninguna otra regla de la tabla, se enviará a la dirección **10.2.7.1** a través de la interfaz **eth207**. Es tu "Puerta de Enlace".
-

2. Rutas Directamente Conectadas

Las siguientes líneas describen redes a las que tu equipo está unido físicamente (o mediante VLANs).

Red de Destino Interfaz (dev) IP de tu equipo (src) Notas

10.2.7.0/24	eth207	10.2.7.3	Red de gestión o tránsito.
172.18.10.0/24	eth217	172.18.10.1	Red local en interfaz 217.
172.18.20.0/24	eth227	172.18.20.1	Red local en interfaz 227.
172.18.30.0/24	eth237	172.18.30.1	Red local en interfaz 237.

Conceptos clave que aparecen:

- **proto kernel:** Significa que la ruta fue creada automáticamente por el sistema operativo al configurar la interfaz. No la puso un administrador manualmente ni un protocolo de enrutamiento dinámico.
- **scope link:** Indica que estas redes son válidas solo en los enlaces directamente conectados; los destinos están en el mismo cable/segmento que tu tarjeta de red.
- **src [IP]:** Es la dirección IP que usará tu máquina como "remitente" por defecto cuando inicie una conexión hacia esa red específica.

Resumen del escenario

Parece que **R1-ubuntu** está actuando como un **router o gateway** para tres subredes distintas (las 172.18.x.x) y utiliza la red 10.2.7.x para comunicarse con el exterior o con un router de nivel superior.

¿Te gustaría que te ayude a configurar una ruta estática adicional o a verificar si el tráfico está fluyendo correctamente entre estas interfaces?

You said

```
10.2.7.0/24 dev eth207 proto kernel scope link src 10.2.7.3
```



Gemini said

Esta línea específica describe una **ruta de red directamente conectada**. Es la forma en que el sistema operativo dice: "Yo pertenezco a esta red y sé cómo hablar con mis vecinos sin ayuda de nadie".

Aquí tienes el desglose técnico de cada término:

- **10.2.7.0/24 (Destino)**: Define el rango de la red. El /24 (máscara de subred 255.255.255.0) indica que esta ruta cubre desde la IP 10.2.7.1 hasta la 10.2.7.254.
- **dev eth207 (Dispositivo)**: Especifica la interfaz física o virtual por la que debe salir el tráfico. En este caso, la tarjeta de red llamada eth207.
- **proto kernel (Protocolo)**: Indica el origen de la ruta. kernel significa que el propio sistema operativo creó esta ruta automáticamente en el momento en que se activó la interfaz con una dirección IP. No fue añadida manualmente por un administrador (static).
- **scope link (Alcance)**: Significa que el destino es local. Los paquetes para esta red se envían directamente al cable/medio físico, ya que se asume que los otros dispositivos están en el mismo segmento de red (no necesitan pasar por un router externo).
- **src 10.2.7.3 (Origen)**: Define la IP que usará tu máquina (**R1-ubuntu**) como dirección de origen cuando envíe paquetes hacia esta red. Es la IP asignada a tu interfaz eth207.

¿Cómo se vería esto en un diagrama de red?

En lenguaje humano:

Tu máquina le está diciendo al sistema: "*Oye, yo tengo la dirección 10.2.7.3. Si alguien quiere enviar algo a cualquier dirección que empiece por 10.2.7.X, que lo mande directamente por el puerto eth207 porque esos equipos están conectados directamente a mí*".

¿Necesitas ayuda para entender cómo se comunica esta red con las otras interfaces 172.18.x.x que aparecen en tu tabla?

You said

tcpdump que es?



Gemini said

tcpdump es una herramienta de línea de comandos cuya función principal es **analizar el tráfico de red**. En el mundo técnico, se le conoce como un "sniffer" o capturador de paquetes.

Imagina que pudieras poner un "escáner de rayos X" en un cable de red para ver exactamente qué datos están pasando por ahí en tiempo real; eso es lo que hace `tcpdump`.

¿Para qué se utiliza?

- **Solucionar problemas:** Si un servicio no funciona, puedes ver si los paquetes realmente están llegando al servidor o si se quedan en el camino.
 - **Seguridad:** Detectar actividad sospechosa, escaneos de puertos o comunicaciones no autorizadas.
 - **Análisis de protocolos:** Ver cómo interactúan las aplicaciones (HTTP, DNS, SSH, etc.) a nivel binario.
 - **Depuración de red:** Verificar si el enrutamiento que vimos antes (como en tu tabla de rutas) está enviando los paquetes por la interfaz correcta.
-

¿Cómo funciona?

`tcpdump` intercepta los paquetes que pasan por una interfaz de red (como tu `eth207`) y muestra una descripción de cada uno. La salida estándar suele verse así:

```
15:30:01.123456 IP 10.2.7.3.54321 > 8.8.8.8.53: 52+ A? google.com. (28)
```

1. **Timestamp:** La hora exacta del paquete.
 2. **Protocolo:** En este caso, IP.
 3. **Origen:** Tu IP (10.2.7.3) y el puerto.
 4. **Destino:** A dónde va (8.8.8.8) y el puerto (53, que es DNS).
 5. **Carga útil:** Qué está preguntando (¿Cuál es la IP de https://www.google.com/url?sa=E&source=gmail&q=google.com?).
-

Comandos básicos que puedes probar

Si quieres ver qué está pasando ahora mismo en tu interfaz `eth207`, podrías usar:

- **Ver tráfico en una interfaz específica:** `tcpdump -i eth207`
- **Ver solo el tráfico de una IP específica:** `tcpdump host 10.2.7.1`
- **Capturar paquetes y guardarlos en un archivo para verlos luego en Wireshark (interfaz gráfica):** `tcpdump -w captura.pcap`
- **Ver solo tráfico HTTP (puerto 80):** `tcpdump port 80`

Nota importante: Para usar `tcpdump` necesitas privilegios de superusuario (`sudo`), ya que poner una interfaz en "modo promiscuo" para leer todo el tráfico es una operación sensible de seguridad.

¿Te gustaría que te ayude a escribir un comando de `tcpdump` específico para verificar si tus redes `172.18.x.x` se están comunicando correctamente con la puerta de enlace?

You said

```
root@R1-ubuntu:~# tcpdump -i eth237
```

```
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
```

```
listening on eth237, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

```
17:09:03.739029 IP 172.18.30.50.53650 > 172.16.200.1.domain: 13867+ A? x.uk. (22)
```

```
17:09:03.739297 IP 172.18.30.50.36605 > 172.16.200.1.domain: 1500+ AAAA? x.uk. (22)
```

```
17:09:03.739975 IP 172.16.200.1.domain > 172.18.30.50.53650: 13867 1/13/12 A 185.249.71.213 (441)
```

```
17:09:03.889593 IP 172.16.200.1.domain > 172.18.30.50.36605: 1500 0/13/12 (425)
```

```
17:09:03.891137 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 1, length 64
```

```
17:09:03.926369 IP 185.249.71.213 > 172.18.30.50: ICMP echo reply, id 3015, seq 1, length 64
```

```
17:09:03.927553 IP 172.18.30.50.40763 > 172.16.200.1.domain: 59041+ PTR? 213.71.249.185.in-addr.arpa. (45)
```

```
17:09:03.928118 IP 172.16.200.1.domain > 172.18.30.50.40763: 59041 NXDomain 0/0/0 (45)
```

```
17:09:04.892284 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 2, length 64
```

```
17:09:04.928347 IP 185.249.71.213 > 172.18.30.50: ICMP echo reply, id 3015, seq 2, length 64
```

```
17:09:05.893080 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 3, length 64
```

```
17:09:05.929114 IP 185.249.71.213 > 172.18.30.50: ICMP echo reply, id 3015, seq 3, length 64
```

```
17:09:06.894742 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 4, length 64
```

```
17:09:06.930652 IP 185.249.71.213 > 172.18.30.50: ICMP echo reply, id 3015, seq 4, length 64
```

```
17:09:07.896175 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 5, length 64
```

```
17:09:07.932351 IP 185.249.71.213 > 172.18.30.50: ICMP echo reply, id 3015, seq 5, length 64
```

```
17:09:08.897976 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 6, length 64
```

```
17:09:08.935281 IP 185.249.71.213 > 172.18.30.50: ICMP echo reply, id 3015, seq 6, length 64
```

```
17:09:09.899938 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 7, length 64
17:09:09.936166 IP 185.249.71.213 > 172.18.30.50: ICMP echo reply, id 3015, seq 7, length 64
17:09:10.901718 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 8, length 64
17:09:10.938006 IP 185.249.71.213 > 172.18.30.50: ICMP echo reply, id 3015, seq 8, length 64
17:09:11.903627 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 9, length 64
17:09:11.939958 IP 185.249.71.213 > 172.18.30.50: ICMP echo reply, id 3015, seq 9, length 64
```



Gemini said

Lo que estás viendo es una **secuencia completa de conectividad exitosa**. Un dispositivo en tu red local está intentando llegar a un sitio web y lo está logrando.

Aquí tienes la "historia" de lo que pasó en esos segundos:

1. La resolución DNS (¿Dónde está "x.uk"?)

Las primeras líneas muestran al equipo con IP **172.18.30.50** preguntando a su servidor DNS (**172.16.200.1**):

- **Petición:** "Oye, ¿cuál es la dirección IP de x.uk?" (registros A y AAAA).
- **Respuesta:** El servidor responde que la IP es **185.249.71.213**.

2. El Ping (ICMP Echo)

Una vez que el equipo sabe la IP, empieza a enviar paquetes de prueba (Pings):

- **ICMP echo request:** Tu equipo local envía el paquete.
- **ICMP echo reply:** El servidor remoto responde.
- **Resultado:** Como ves ambos (request y reply), la conexión es bidireccional y saludable. El tiempo de respuesta (latencia) es de unos **36ms** aproximadamente (la diferencia entre el .891 y el .926).

Análisis de los protagonistas:

- **El Cliente (172.18.30.50):** Un host conectado a tu interfaz eth237.
- **El Servidor DNS (172.16.200.1):** El encargado de traducir nombres a IPs.
- **El Destino (185.249.71.213):** El servidor externo al que se le está haciendo el ping.

Un detalle curioso: NXDomain

En la línea 17:09:03.928118, el servidor DNS responde con **NXDomain** a una consulta de tipo PTR (búsqueda inversa).

- **Traducción:** El equipo 172.18.30.50 intentó preguntar "¿Cómo se llama el equipo que tiene la IP 185.249.71.213?", pero el servidor DNS no tiene ese nombre registrado y respondió "No existe ese dominio". Es normal y no afecta al ping.
-

¿Qué significa esto para ti?

Tu configuración de red en **R1-ubuntu** está funcionando correctamente como router para la red 172.18.30.0/24:

1. Recibe el tráfico por eth237.
2. Lo encamina hacia el exterior (usando la ruta default que vimos antes).
3. Permite que las respuestas regresen al cliente.

¿Quieres intentar filtrar el tráfico para ver solo los errores o prefieres ver qué está pasando en las otras interfaces (como la eth227)?

You said

tcpdump -vvv



Gemini said

Cuando ejecutas `tcpdump -vvv`, le estás pidiendo al sistema el **máximo nivel de detalle posible** (el modo "triple verbose").

En este modo, `tcpdump` no solo te dice quién habla con quién, sino que abre el paquete y te muestra las "letras chiquitas" de las cabeceras de red.

¿Qué información extra verás con -vvv?

Al usar el triple nivel de detalle, la salida se vuelve mucho más densa. Aquí tienes lo que se añade:

1. **TTL (Time to Live):** Cuántos "saltos" (routers) le quedan al paquete antes de ser descartado. Útil para detectar bucles de red.
2. **ID de IP:** Un número de identificación único para cada paquete, clave para rastrear fragmentación.

3. **Flags de TCP:** Detalles específicos como [S] (Sync), [.] (Ack), [P] (Push), que ayudan a entender en qué fase está una conexión.
 4. **Checksums:** Verificaciones de errores para asegurar que el paquete no se corrompió en el cable.
 5. **Opciones de IP/TCP:** Parámetros como el tamaño de la ventana (win), el MSS (Maximum Segment Size) y timestamps de SACK.
-

Ejemplo de cómo cambia la salida

Sin -vvv (Normal): 17:09:03 IP 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 1

Con -vvv: 17:09:03.891137 IP (tos 0x0, ttl 64, id 12345, offset 0, flags [DF], proto ICMP (1), length 84) 172.18.30.50 > 185.249.71.213: ICMP echo request, id 3015, seq 1, length 64

¿Cuándo usar cada nivel?

- **-v:** Información básica (TTL, ID, longitud total).
- **-vv:** Añade detalles de protocolos específicos (por ejemplo, respuestas NFS o detalles de SMB).
- **-vvv:** El análisis más profundo. Útil cuando sospechas que un router está modificando los paquetes o cuando hay problemas de fragmentación.

Tip Pro: Combínalo con -n y -e

Si usas -vvv, la pantalla se llenará de texto muy rápido. Te recomiendo añadir estos parámetros:

- **-n:** No resuelvas nombres (muestra IPs en lugar de dominios). Es **mucho más rápido** y evita generar más tráfico DNS.
- **-e:** Muestra también la **dirección MAC** de origen y destino. Vital si quieres saber si el problema es de Capa 2 (Ethernet).

```
tcpdump -i eth237 -nnvvve
```

¿Quieres que analicemos algún error específico que te esté apareciendo ahora con ese nivel de detalle?

You said

```
17:11:48.330585 IP (tos 0x10, ttl 64, id 60403, offset 0, flags [DF], proto TCP (6), length 548)
```

```
R1-ubuntu.institutodh.net.ssh > 10.8.3.96.9238: Flags [P.], cksum 0x2083 (incorrect -> 0xb5c1), seq 95280:95788, ack 1549, win 625, length 508
```

```
17:11:48.394042 IP (tos 0x0, ttl 126, id 39358, offset 0, flags [DF], proto TCP (6), length 40)
```

10.8.3.96.9238 > R1-ubuntu.institutodh.net.ssh: Flags [.], cksum 0x0591 (correct), seq 1549, ack 95788, win 252, length 0

17:11:48.434587 IP (tos 0x10, ttl 64, id 60404, offset 0, flags [DF], proto TCP (6), length 548)

R1-ubuntu.institutodh.net.ssh > 10.8.3.96.9238: Flags [P.], cksum 0x2083 (incorrect -> 0x4943), seq 95788:96296, ack 1549, win 625, length 508

17:11:48.507935 IP (tos 0x0, ttl 126, id 39359, offset 0, flags [DF], proto TCP (6), length 40)

10.8.3.96.9238 > R1-ubuntu.institutodh.net.ssh: Flags [.], cksum 0x0392 (correct), seq 1549, ack 96296, win 255, length 0

17:11:48.537587 IP (tos 0x10, ttl 64, id 60405, offset 0, flags [DF], proto TCP (6), length 548)

R1-ubuntu.institutodh.net.ssh > 10.8.3.96.9238: Flags [P.], cksum 0x2083 (incorrect -> 0x8c27), seq 96296:96804, ack 1549, win 625, length 508

17:11:48.603113 IP (tos 0x0, ttl 126, id 39360, offset 0, flags [DF], proto TCP (6), length 40)

10.8.3.96.9238 > R1-ubuntu.institutodh.net.ssh: Flags [.], cksum 0x0197 (correct), seq 1549, ack 96804, win 254, length 0

17:11:48.642584 IP (tos 0x10, ttl 64, id 60406, offset 0, flags [DF], proto TCP (6), length 548)

R1-ubuntu.institutodh.net.ssh > 10.8.3.96.9238: Flags [P.], cksum 0x2083 (incorrect -> 0x3c7f), seq 96804:97312, ack 1549, win 625, length 508

17:11:48.714497 IP (tos 0x0, ttl 126, id 39361, offset 0, flags [DF], proto TCP (6), length 40)

10.8.3.96.9238 > R1-ubuntu.institutodh.net.ssh: Flags [.], cksum 0xff9c (correct), seq 1549, ack 97312, win 252, length 0

17:11:48.746588 IP (tos 0x10, ttl 64, id 60407, offset 0, flags [DF], proto TCP (6), length 548)

R1-ubuntu.institutodh.net.ssh > 10.8.3.96.9238: Flags [P.], cksum 0x2083 (incorrect -> 0x633b), seq 97312:97820, ack 1549, win 625, length 508

17:11:48.810596 IP (tos 0x0, ttl 126, id 39362, offset 0, flags [DF], proto TCP (6), length 40)

10.8.3.96.9238 > R1-ubuntu.institutodh.net.ssh: Flags [.], cksum 0xfd9d (correct), seq 1549, ack 97820, win 255, length 0



Gemini said

Esta captura es fascinante porque muestra una **sesión de SSH activa** (probablemente tu propia conexión a la máquina) y revela un fenómeno técnico muy común en redes modernas.

Aquí tienes el análisis de lo que está ocurriendo:

1. Los Protagonistas

- **Origen:** R1-ubuntu (tu servidor) enviando datos desde el puerto **22 (SSH)**.
 - **Destino:** 10.8.3.96 en el puerto **9238**. Este es el cliente (tu computadora) que está controlando la terminal.
 - **Actividad:** Estás viendo paquetes con el flag **[P.]** (Push-Acknowledgment). Esto significa que el servidor está enviando datos (probablemente el texto del propio tcpdump que estás ejecutando) y el cliente responde con un **[.]** (Ack) para confirmar que los recibió.
-

2. El "Misterio" del Checksum Incorrecto

Verás que en los paquetes que salen de R1-ubuntu dice: cksum 0x2083 (incorrect -> 0xb5c1)

¿Significa que tu red está rota? **No**. Esto se debe a una característica llamada **TCP Checksum Offloading**.

- **Explicación:** Para ahorrar trabajo al procesador (CPU), el sistema operativo le delega la tarea de calcular el código de error (checksum) a la tarjeta de red (NIC) justo antes de enviar el paquete al cable.
 - **Por qué tcpdump dice "incorrect":** Como tcpdump captura el paquete cuando sale de la CPU pero **antes** de que llegue a la tarjeta de red, el campo del checksum todavía está vacío o tiene basura. La tarjeta de red lo corregirá un milisegundo después, pero tcpdump ya no lo ve.
-

3. Detalles de los Campos Verbos (-vvv)

Gracias al nivel de detalle, podemos ver parámetros avanzados:

- **tos 0x10:** Es el tipo de servicio (Type of Service). El valor 0x10 indica "**Low Delay**" (baja latencia), que es el estándar para conexiones interactivas como SSH.
 - **ttl 64 vs 126:** * Tu servidor sale con **TTL 64** (estándar de Linux).
 - El cliente llega con **TTL 126**. Como Windows suele empezar en 128, esto sugiere que el cliente es una máquina Windows y hay 2 routers de distancia entre tú y ella.
 - **length 548:** Es el tamaño del paquete. SSH envía los datos en bloques para ser eficiente.
 - **win 625:** Es el "Window Size". Indica cuánto espacio queda en el búfer para seguir recibiendo datos sin necesidad de pausar.
-

Resumen de la situación

Lo que estás viendo es **el eco de tu propia sesión**. Cada vez que tcpdump imprime una línea en tu pantalla, genera un paquete de red para enviarte ese texto por SSH, lo cual genera otra línea en tcpdump, creando un bucle infinito de tráfico (por eso ves tantos paquetes seguidos).

Tip: Si quieres usar tcpdump sin que tu propia sesión de SSH estorbe y llene la pantalla, usa este filtro para excluir el puerto 22: `tcpdump -i any -vvv not port 22`

¿Te gustaría que probemos este filtro para ver si hay algún otro tráfico "escondido" en tus interfaces?

C