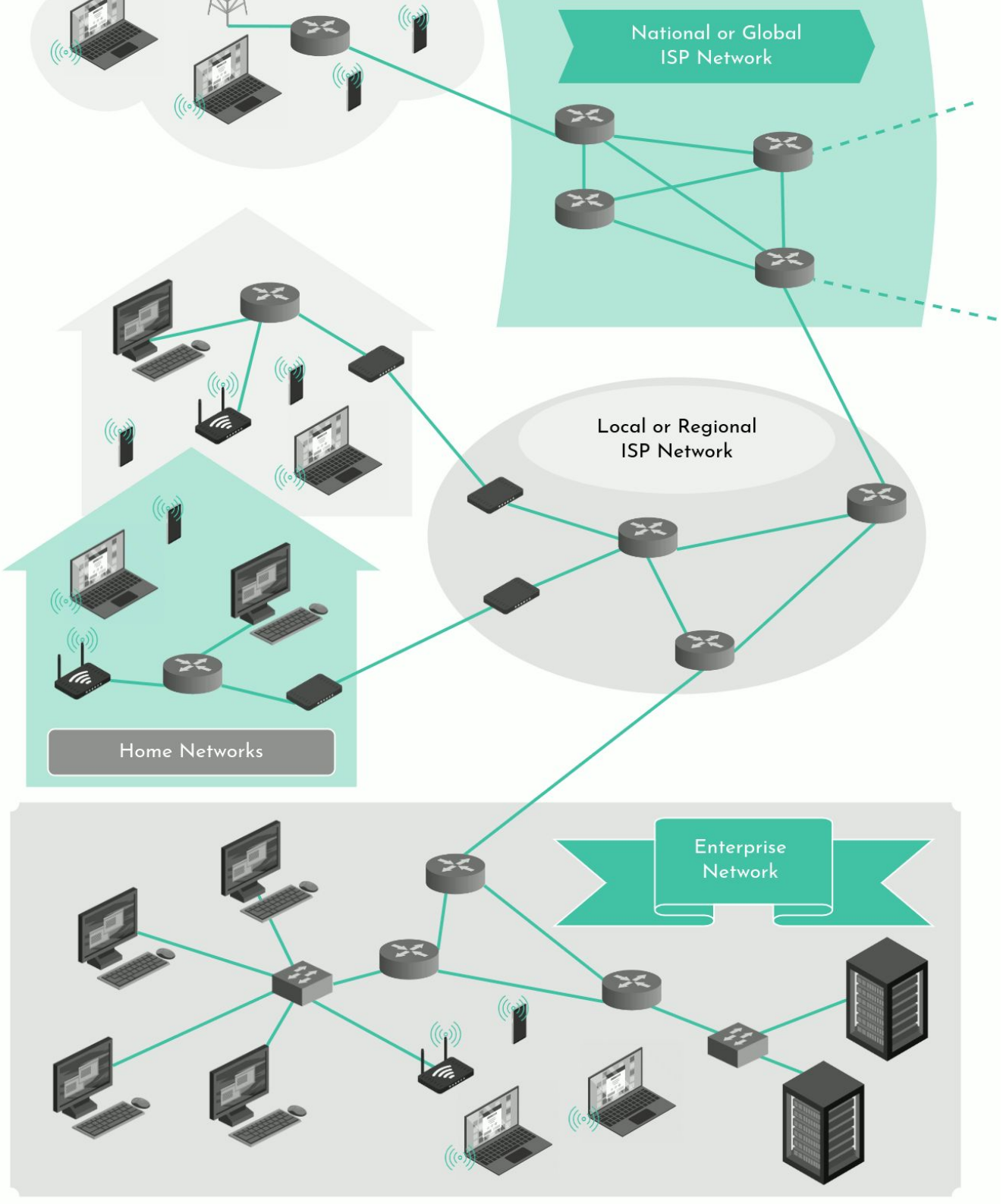

**Administración y Gestión de Redes**
Lic. en Sistemas de Información

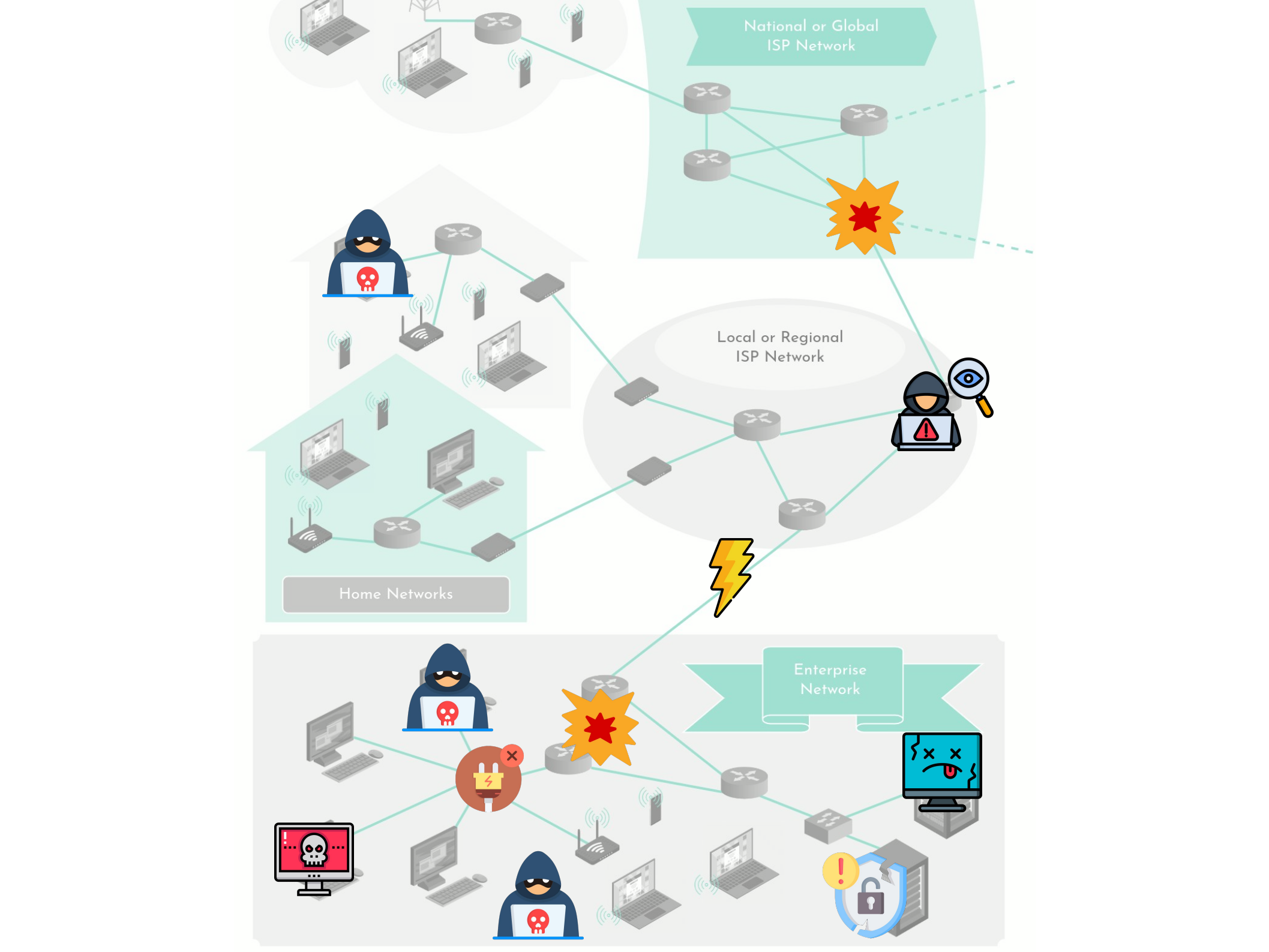
**Laboratorio de REDES**
Recuperación de Información
y Estudios de la Web

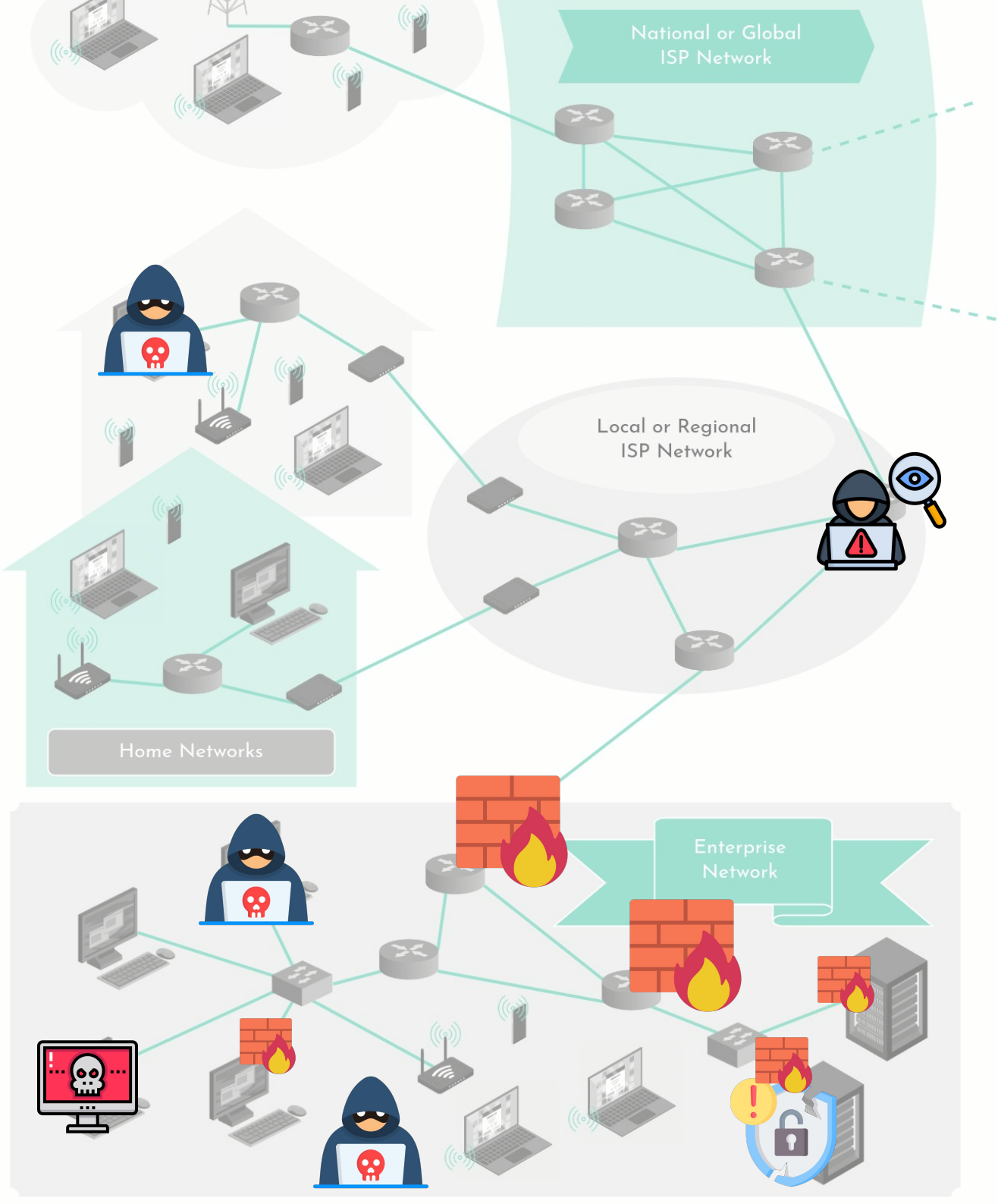
# Firewalls - Netfilter

Equipo docente:
Santiago Ricci <sricci@unlu.edu.ar>
Alejandro Iglesias <aaiglesias@unlu.edu.ar>
Mauro Meloni <maurom@unlu.edu.ar>
Fernando Lorge <florge@unlu.edu.ar>

**- 2026 -**


National or Global
ISP Network
Local or Regional
ISP Network
Home Networks
Enterprise
Network


National or Global
ISP Network
Local or Regional
ISP Network
Home Networks
Enterprise
Network


National or Global
ISP Network
Local or Regional
ISP Network
Home Networks
Enterprise
Network

# Netfilter

Netfilter es un framework extensible dentro del kernel de Linux que permite realizar diversas acciones sobre los paquetes que atraviesan el stack de networking del sistema operativo.

**Mediante netfilter es posible...**

- implementar filtrado de paquetes sin estado,
- implementar filtrado de paquetes con estado,
- realizar traducción de direcciones de red y puertos (NAT/NATP) en sus múltiples variantes,
- asistir al framework ***tc*** para construir políticas complejas de QoS,
- modificar paquetes (*mangling*) de diversas formas, por ej. alterar los bits TOS/DSCP/ECN del encabezado IP.

Los comandos más conocidos de netfilter son...

# iptables y nft

Los comandos ***iptables*** y su reemplazo más reciente ***nft*** se utilizan para configurar, mantener e inspeccionar las tablas de reglas de filtro de paquetes IPv4 e IPv6 en el kernel de Linux.

Iptables provee por defecto tres tablas (**TABLES**) donde agregar reglas: las tablas ***filter***, ***nat*** y ***mangle***. Las primeras dos tablas son evidentes, la última sirve para modificar campos particulares y marcar paquetes.

Cada tabla contiene una serie de cadenas (**CHAINS**) predefinidas (que en breve veremos) y otras definidas por el usuario.

Cada cadena es una lista de reglas (**RULES**) que pueden coincidir con uno o más paquetes. Cada regla especifica qué acción (**VERDICT**) tomar con un paquete que coincide.

# La vida de un paquete en el S.O.

- Paquete entrante, **destinado a este equipo**.

- Paquete saliente, **enviado** desde el equipo.

- Paquete entrante, **NO destinado** a este equipo.

Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

# La vida de un paquete en el S.O.

- Paquete entrante, **destinado a este equipo**.
  - ¿Cómo saber si el paquete está destinado al host?

- Paquete saliente, **enviado** desde el equipo.
  - Fácil de reconocer.
- Paquete entrante, **NO destinado** a este equipo.
  - ¿En qué caso o casos veríamos uno de estos?

Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

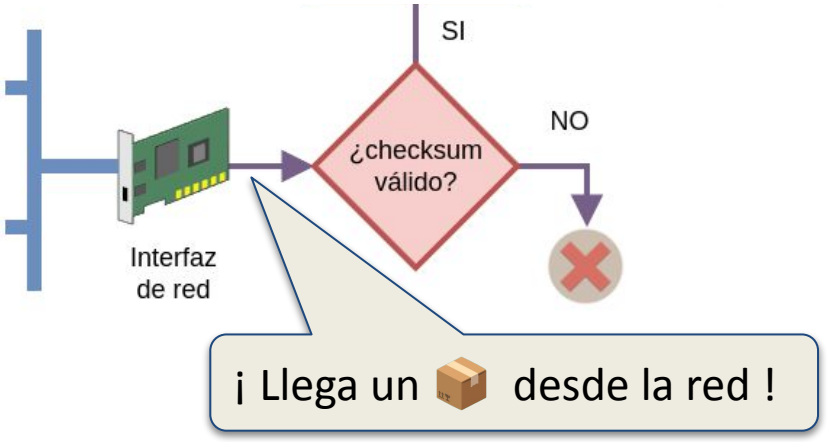
# La vida de un paquete en el S.O.

- Paquete entrante, **destinado a este equipo**.
  - ¿Cómo saber si el paquete está destinado al host?
    - Dirección IP destino COINCIDE con la actual.
- Paquete saliente, **enviado** desde el equipo.
  - Fácil de reconocer.
- Paquete entrante, **NO destinado** a este equipo.
  - ¿En qué caso o casos veríamos uno de estos?
    - dir IP origen **Y** dir IP destino **NO** propias.

Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

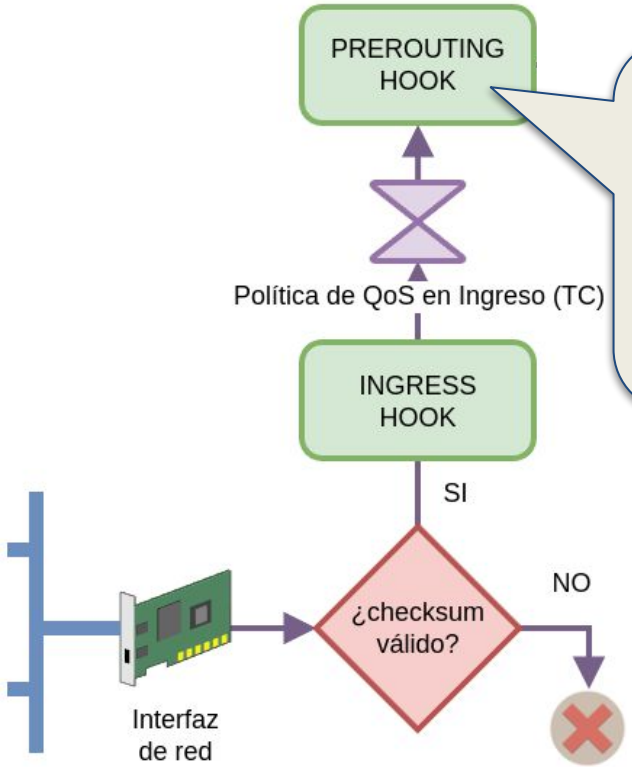
# La vida de un paquete en el S.O.

- Paquete entrante, **destinado a este equipo**.
  - ¿Cómo saber si el paquete está destinado al host?
    - Dirección IP destino COINCIDE con la actual.




Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

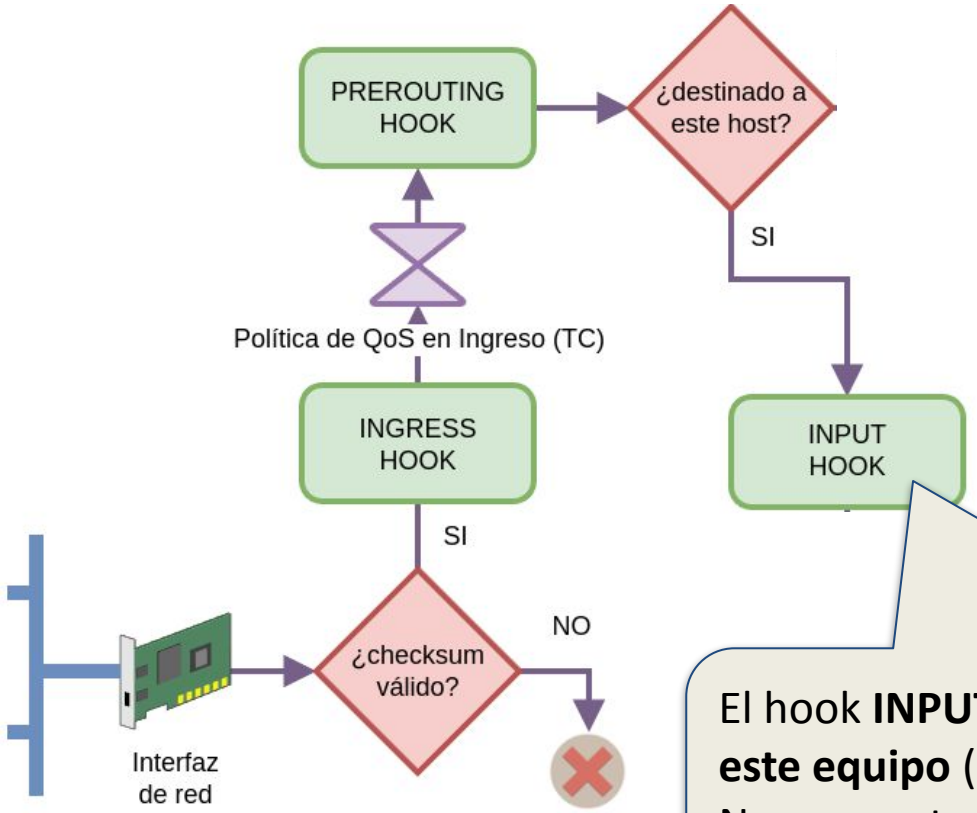
# La vida de un paquete en el S.O.




El hook **PREROUTING** es disparado para TODOS los paquetes que llegan desde la red, ANTES de que se tome la decisión de ruteo. Cada ***cadena o chain*** asociada a un ***hook*** puede tener cero, una o más reglas que se aplican a los paquetes que la atraviesan (y que cumplen las condiciones indicadas en cada regla).

Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

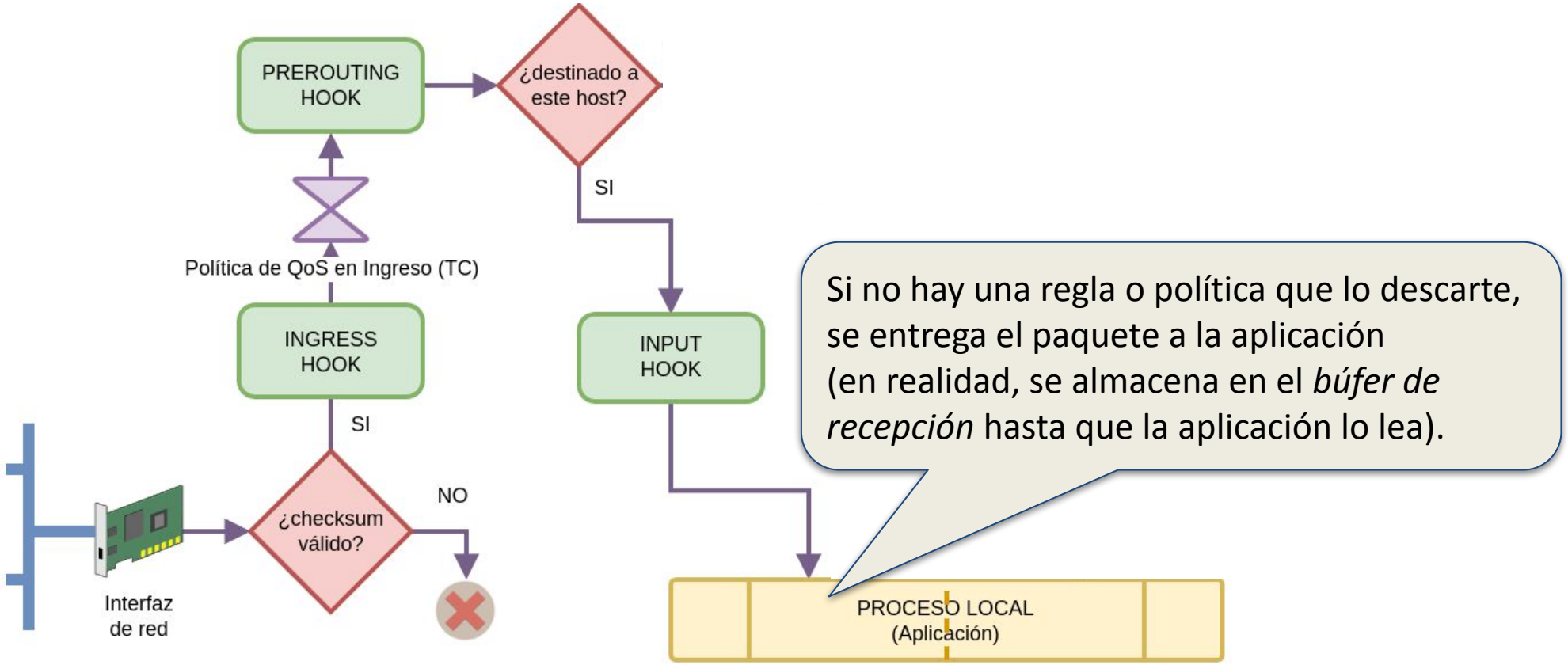
# La vida de un paquete en el S.O.




El hook **INPUT** se dispara por cada paquete **DESTINADOS a este equipo** (dirección IP destino = host).
Nuevamente, las cadenas asociadas pueden contener cero, una o más reglas de filtrado o modificación de paquetes.

Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

# La vida de un paquete en el S.O.




Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

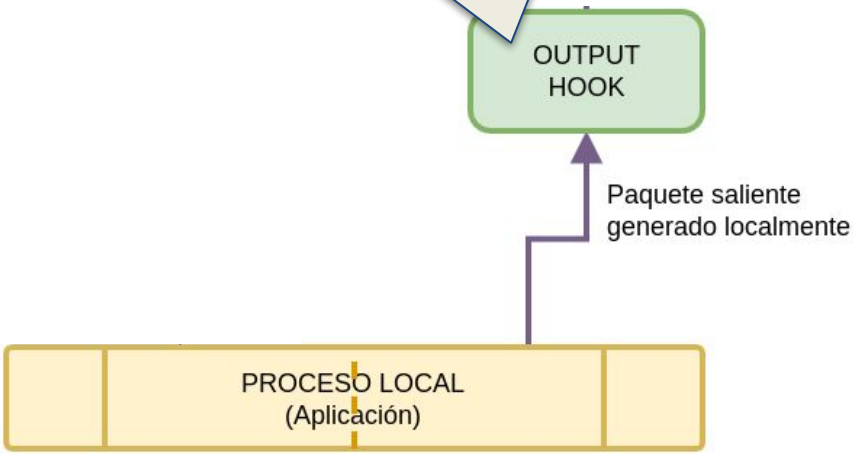
# La vida de un paquete en el S.O.

- Paquete saliente, **enviado** desde el equipo.
  - Fácil de reconocer.

Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

# La vida de un paquete en el S.O.

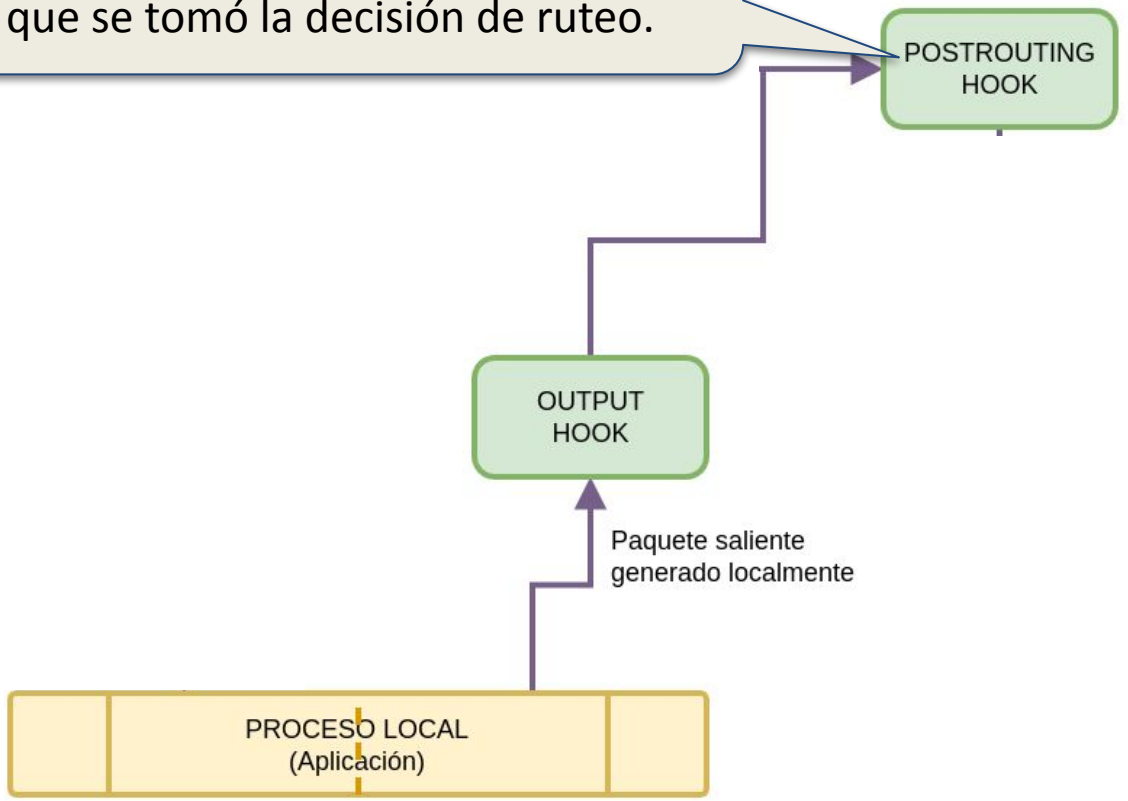
Supongamos que la aplicación leyó el paquete e intenta enviar una respuesta (o bien la capa TCP intenta enviar un ACK). Los paquetes **ORIGINADOS** en este host (dirección IP origen = host) disparan un evento en el hook **OUTPUT**.




Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

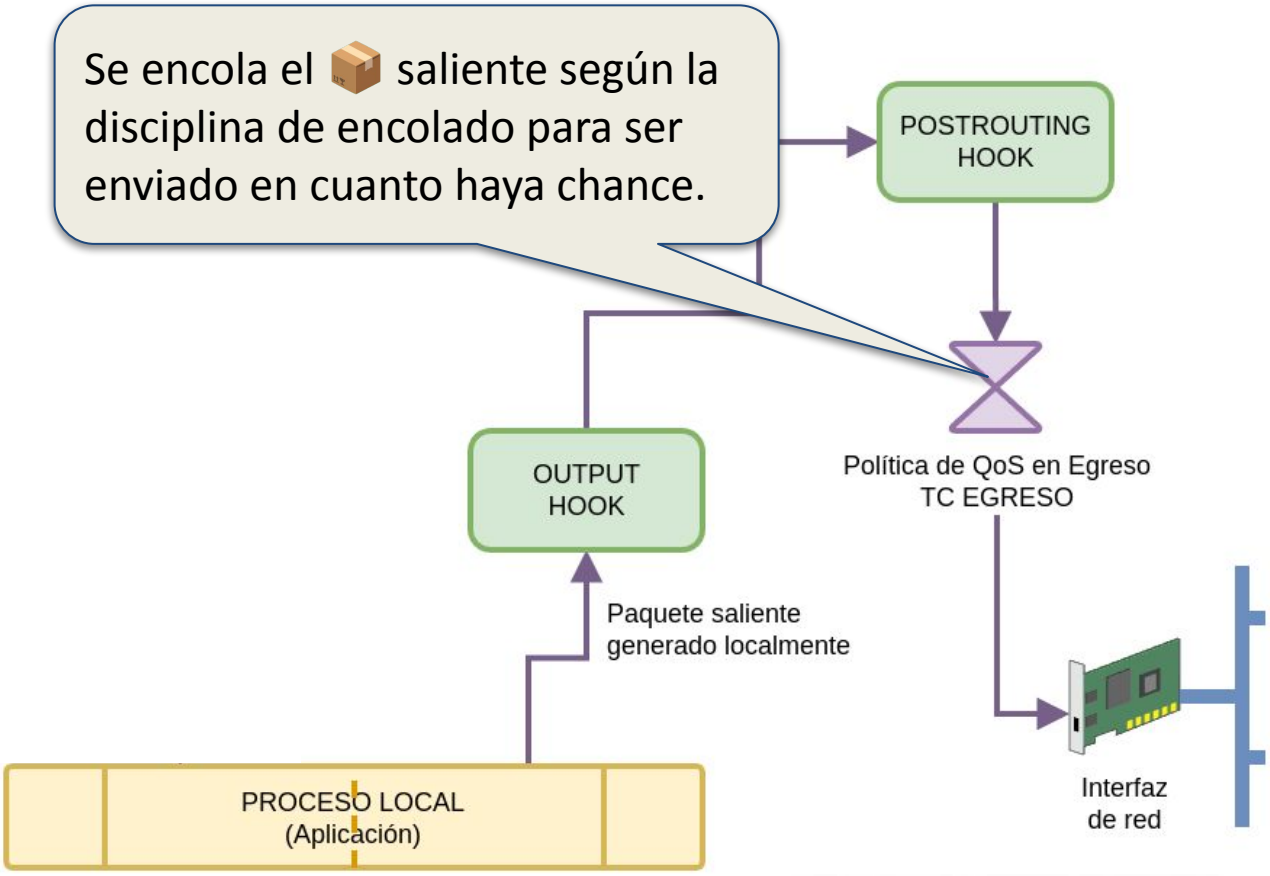
# La vida de un paquete en el S.O.

El hook **POSTROUTING** se dispara para TODOS los paquetes que salen hacia la red, DESPUÉS de que se tomó la decisión de ruteo.




Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

# La vida de un paquete en el S.O.



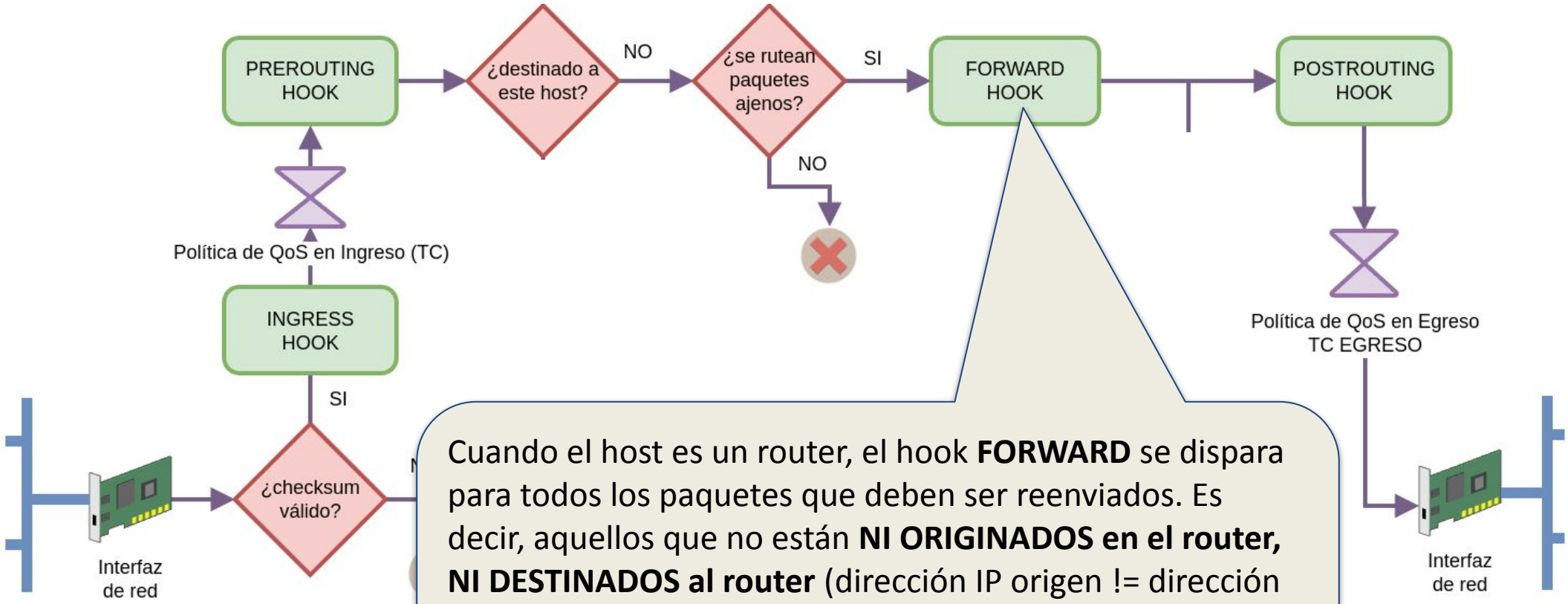

Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

# La vida de un paquete en el S.O.

- Paquete entrante, **NO destinado** a este equipo.
  - ¿En qué caso o casos veríamos uno de estos?
    - dir IP origen **Y** dir IP destino **NO** propias.

Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

# La vida de un paquete en el S.O.




Cuando el host es un router, el hook **FORWARD** se dispara para todos los paquetes que deben ser reenviados. Es decir, aquellos que no están **NI ORIGINADOS en el router, NI DESTINADOS al router** (dirección IP origen != dirección IP destino != dirección IP del router).
Aquí es donde se coloca la mayoría de las reglas de filtrado de un firewall de red.

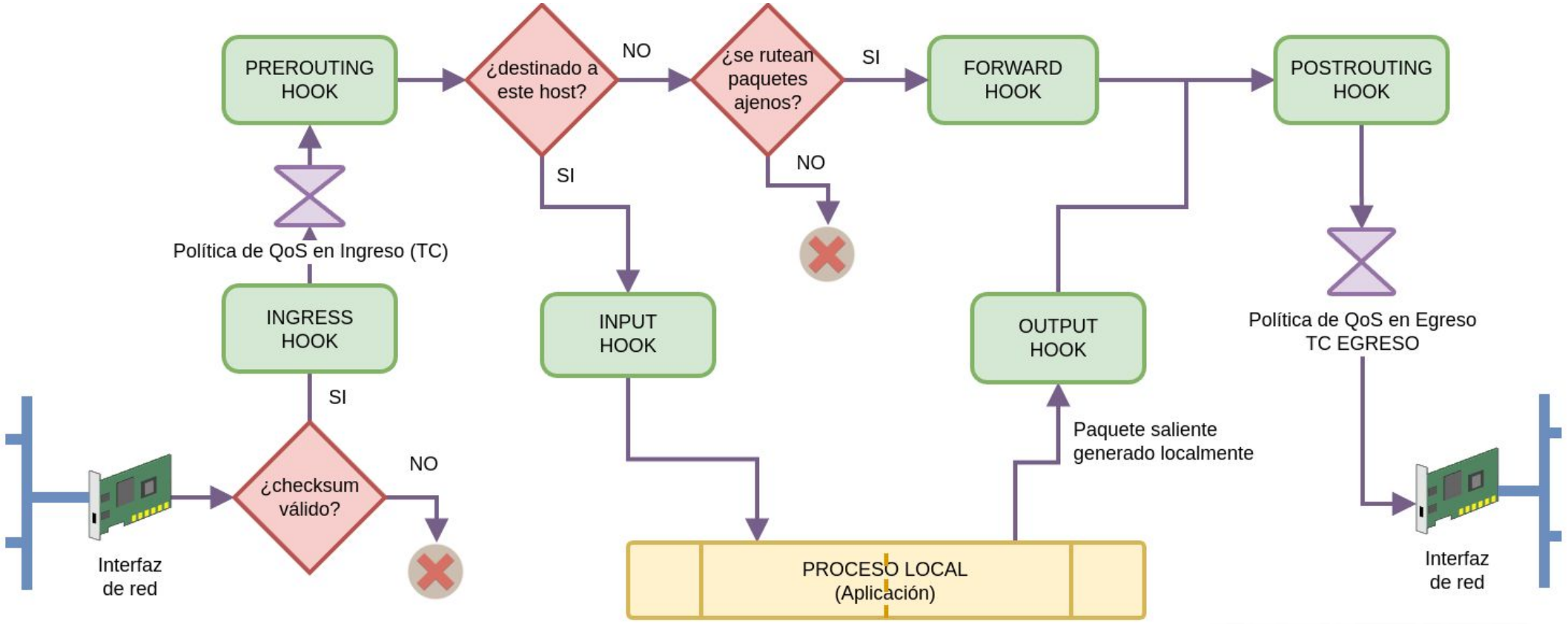
Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

# La vida de un paquete en el S.O.

- Paquete entrante, **destinado a este equipo**.
  - ¿Cómo saber si el paquete está destinado al host?
    - Dirección IP destino COINCIDE con la actual.
- Paquete saliente, **enviado** desde el equipo.
  - Fácil de reconocer.
- Paquete entrante, **NO destinado** a este equipo.
  - ¿En qué caso o casos veríamos uno de estos?
    - dir IP origen **Y** dir IP destino **NO** propias.

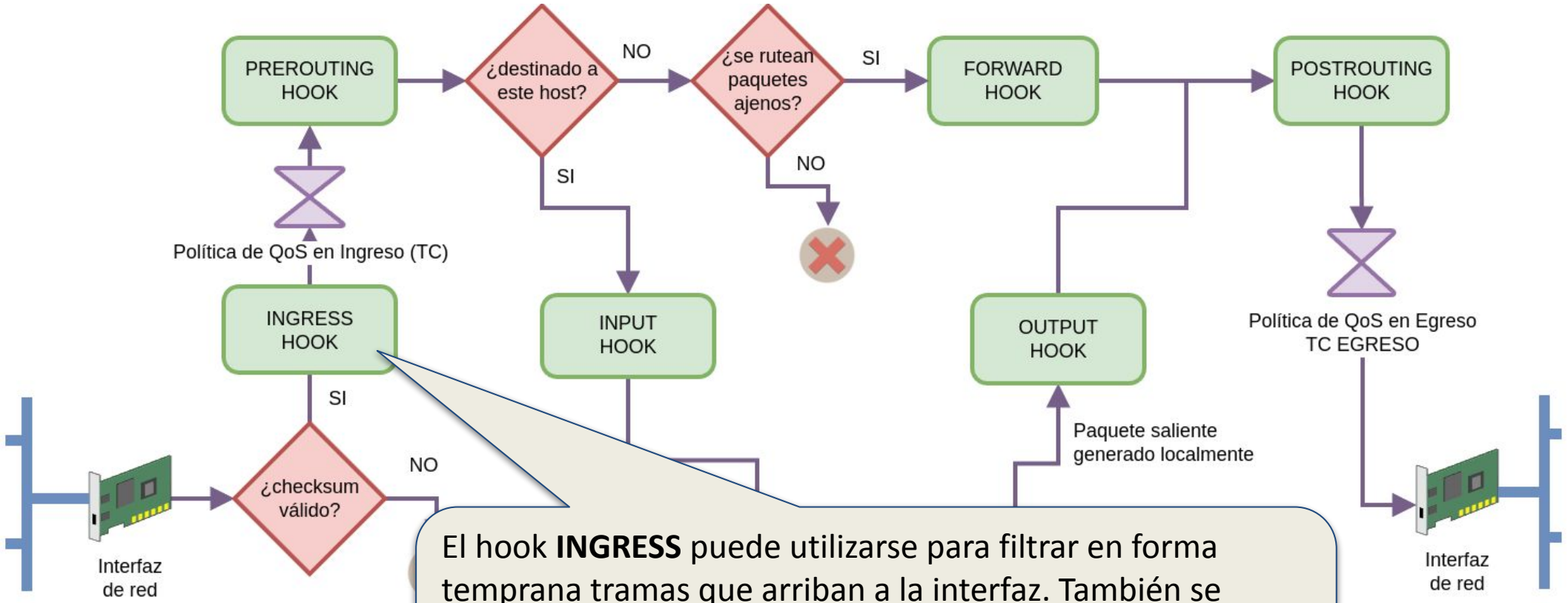
Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

# La vida de un paquete en el S.O.




Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

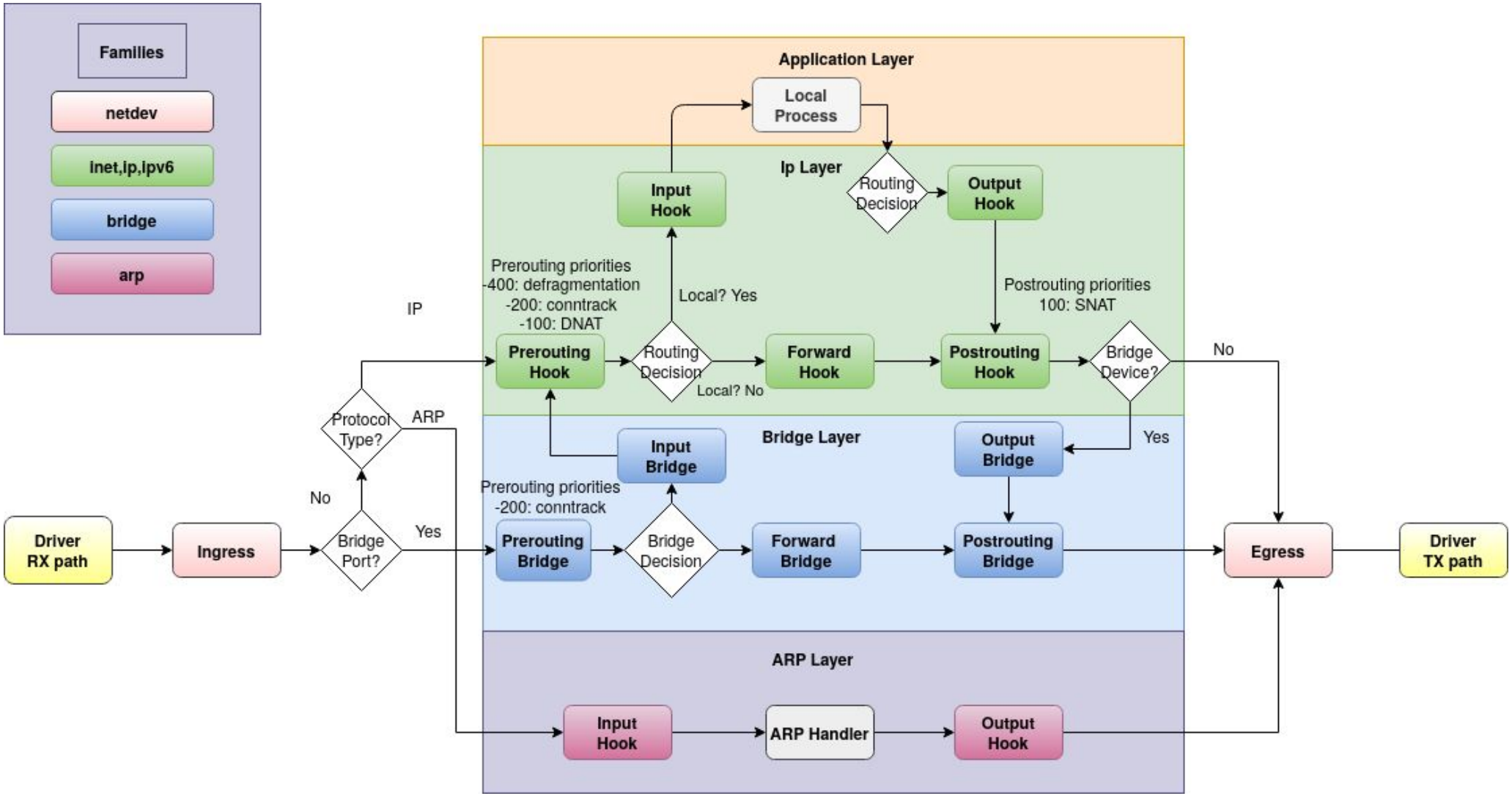
# La vida de un paquete en el S.O.




El hook **INGRESS** puede utilizarse para filtrar en forma temprana tramas que arriban a la interfaz. También se pueden filtrar paquetes, pero las reglas se aplicarán antes de que estén reensamblados (si venían fragmentados). En tal caso, ¡cuidado al filtrar por puertos TCP/UDP!

Para un gráfico más completo, ver https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks

# La vida de un paquete en el S.O.



# Reglas en netfilter

Cada cadena contiene una lista de reglas (similar a una **ACL**) y una política por defecto (*accept* o *drop*) si ninguna regla coincide con un paquete.

Cada regla de firewall especifica los **criterios o condiciones** al evaluar un paquete y una acción o **VERDICT** que efectuar si el paquete las cumple.

Si el paquete no coincide, se examina la siguiente regla de la cadena;

Si coincide, se ejecuta lo indicado en “acción”, pudiendo ser:

- uno de los valores especiales
  **ACCEPT, DROP, RETURN, REJECT** o **LOG**,
- uno de los objetivos descritos en ***iptables-extensions***,
- un salto a otra cadena de reglas definida por usuario

# Acciones predefinidas

- **ACCEPT**: permite al paquete seguir su camino.
- **REJECT**: rechaza el paquete con un paquete de error ICMP (la opción `--reject-with` de iptables permite seleccionar el tipo de error icmp).
- **DROP**: descarta el paquete sin generar ningún mensaje ni error.
- **LOG**: registra (en el log del sistema) un mensaje con una descripción del paquete; esta regla no interrumpe el procesamiento y habitualmente requiere de una segunda regla que defina qué se hace.
- **RETURN**: interrumpe el procesamiento de la cadena actual y regresa a la cadena que llamó a ésta (si existiera).

… hay muchas más en ***man iptables-extensions***

# Ejemplo de Reglas (ACL)

INPUT HOOK CHAIN en un firewall de red -- Default policy: ¿cuál?

| # | IP ORIGEN | IP DESTINO | PROTO | PUERTO ORIG. | PUERTO DEST. | ACCIÓN |
|---|---|---|---|---|---|---|
| 1 | * (cualquiera) | 170.210.96.3 | TCP | > 1024 | 22 | ACCEPT |
| 2 | 170.210.96.73 | 170.210.96.3 | UDP | * | 161 | ACCEPT |
| 3 | 170.210.96.0/24 | 170.210.96.3 | ICMP | - | - | ACCEPT |
| | ... | ... | | ... | ... | |

FORWARD HOOK CHAIN en un firewall de red -- Default policy: ¿cuál?

| # | IP ORIGEN | IP DESTINO | PROTO | PUERTO ORIG. | PUERTO DEST. | ACCIÓN |
|---|---|---|---|---|---|---|
| 1 | * (cualquiera) | 170.210.96.1 | TCP | * | 80 | ACCEPT |
| 2 | 170.210.96.1 | * | TCP | 80 | * | ACCEPT |
| 3 | * | 170.210.96.1 | TCP | * | 22 | LOG |
| 4 | 200.110.185.0/24 | * | TCP | * | * | DROP |

# Ejemplo de Reglas (ACL)

OUTPUT HOOK CHAIN en un firewall de red -- Default policy: ¿cuál?

| # | IP ORIGEN | IP DESTINO | PROTO | PUERTO ORIG. | PUERTO DEST. | ACCIÓN |
|---|---|---|---|---|---|---|
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| | ... | ... | | ... | ... | |

# Similitud con Access Control Lists

INPUT HOOK CHAIN en un servidor de correo -- Default policy: ¿cuál?

| # | IP ORIGEN | IP DESTINO | PROTO | PUERTO ORIG. | PUERTO DEST. | ACCIÓN |
|---|---|---|---|---|---|---|
| 1 | * (cualquiera) | 170.210.96.3 | TCP | > 1024 | 25 | ACCEPT |
| 2 | 170.210.96.73 | 170.210.96.3 | UDP | * | 161 | ACCEPT |
| 3 | 170.21 6.0/24 | 17 10.96.3 | ICMP | - | | ACC T |
| | | ... | | ... | | |

Actor

Recurso

Recurso

"Permiso"

FORWARD HOOK CHAIN en un firewall de red -- Default policy: ¿cuál?

| # | IP ORIGEN | IP DESTINO | PROTO | PUERTO ORIG. | PUERTO DEST. | ACCIÓN |
|---|---|---|---|---|---|---|
| 1 | * (cualquiera) | 170.210.96.1 | TCP | * | 80 | ACCEPT |
| 2 | 170.210.96.1 | * | TCP | 80 | * | ACCEPT |
| 3 | * | 170.210.96.1 | TCP | * | 22 | LOG |
| 4 | 200.110.185.0/24 | * | TCP | * | * | DROP |

# Reglas de ida y vuelta

INPUT HOOK CHAIN en un servidor de correo -- Default policy: ¿cuál?

| # | IP ORIGEN | IP DESTINO | PROTO | PUERTO ORIG. | PUERTO DEST. | ACCIÓN |
|---|---|---|---|---|---|---|
| 1 | * (cualquiera) | 170.210.96.3 | TCP | > 1024 | 25 | ACCEPT |
| 2 | 170.210.96.73 | 170.210.96.3 | UDP | * | 161 | ACCEPT |
| 3 | 170.210.96.0/24 | 170.210.96.3 | ICMP | - | - | ACCEPT |
| | ... | ... | | ... | ... | |

FORWARD HOOK CHAIN en un firewall de red -- Default policy: ¿cuál?

deja pasar la PETICIÓN al webserver »

| # | IP ORIGEN | IP DESTINO | PROTO | PUERTO ORIG. | PUERTO DEST. | ACCIÓN |
|---|---|---|---|---|---|---|
| 1 | * (cualquiera) | 170.210.96.1 | TCP | * | 80 | ACCEPT |
| 2 | 170.210.96.1 | * | TCP | 80 | * | ACCEPT |
| 3 | * | 170.210.96.1 | TCP | * | 22 | LOG |
| 4 | 200.110.189.0/24 | * | TCP | * | * | DROP |

« deja pasar la RESPUESTA del webserver

¿Qué tipo de firewall es? ¿Cómo puede mejorarse?

# Sintaxis de nft

- Primero hay que crear una tabla y al menos una cadena

  `nft add table inet FW && nft add chain inet FW INPUT`

- Las reglas se agregan/eliminan/listan con la siguiente sintaxis:

  `nft add rule inet TABLA CADENA REGLA-DE-MATCHEO ACCION`

- por ejemplo, para agregar una regla que rechace todos los paquetes IP destinados al puerto 80 del firewall, el comando a ejecutar es:

  `nft add rule FW INPUT ip daddr IP-DEL-FW tcp dport 80 reject`

- El primer argumento indica el comando a ejecutar:

  list listar las reglas en una cadena (verbose y sin resolver DNS)
  add agregar la regla indicada a continuación
  delete borrar la regla indicada mediante un número de regla
  flush vaciar (limpiar) toda la lista de reglas de una cadena

# Especificación de reglas

- Las “condiciones” que se definen en una regla pueden ser múltiples.
- Las más habituales son:

| | |
|---|---|
| `ip protocol YYY` | coincide con el campo de protocolo “de transporte”, los valores más comunes son *tcp, udp, icmp* e *icmpv6*. |
| `ip saddr IP_ADDR` | coincide con la dirección de host origen (o red si indica máscara) |
| `ip daddr IP_ADDR` | coincide con la dirección de host destino (o red si indica máscara) |
| `iif INTERFAZ` | coincide con la interfaz de red donde arribó un paquete |
| `tcp sport NN` | coincide con el número de puerto TCP o UDP origen |
| `tcp dport NN` | coincide con el número de puerto TCP o UDP destino |
| `ct state new` | coincide con paquetes TCP de apertura de conexión |
| `ct state ...` | coincide con un estado de conexión particular (para stateful) |

- Es posible negar una condición con el signo **!**
- Montones de ejemplos en la práctica...

# Bibliografía


- STALLINGS, W. 2011. *Cryptography and Network Security: Principles and Practice* (5th ed). Prentice Hall.
  - Capítulo 1: Overview
  - Capítulo 22: Firewalls
- HERTZOG, R. & MAS, R. 2015. *El manual del Administrador de Debian*. Freexian.
  - Capítulo 14. Sección 2: "Firewall o el filtrado de paquetes"
- ZWICKY, E.; COOPER S. & CHAPMAN D. B.. 2000. *Building Internet Firewalls* (2nd ed). O'Reilly Media.
  - Capítulo 3: Security Strategies
- EVANS, Julia - iptables. https://twitter.com/b0rk/status/1054056111626686465


**Próxima: Criptografía**