Riassunto della rete virtuale

1.1 La rete di Qemu

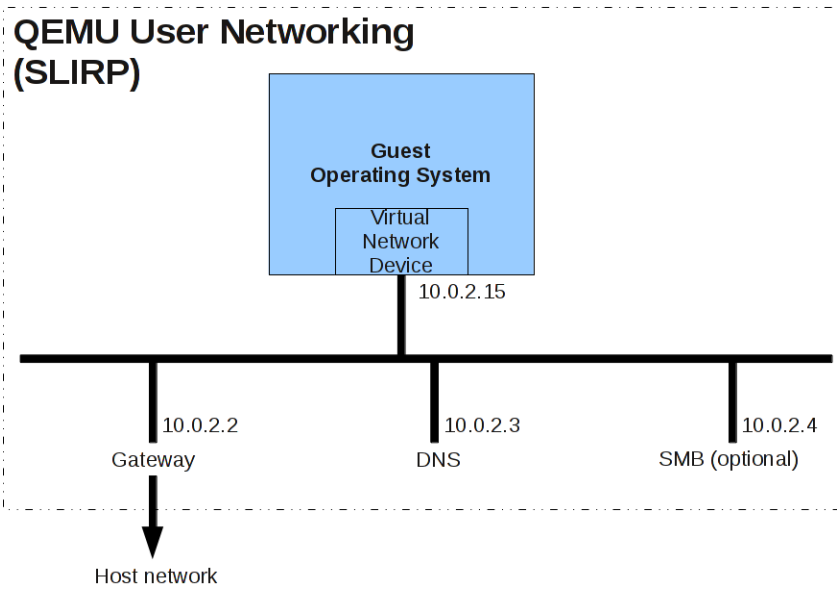

Figura 1.1: La rete virtuale creata da Qemu in modalità user networking

Qemu implementa una rete virtuale minimale con la topologia descritta nella figura 1.1. La macchina *guest* è un nodo che generalmente ha il numero IP `10.0.2.15` (assegnato tramite DHCP dal nodo `10.0.2.2`, che svolge anche le funzioni di *gateway*), anche se naturalmente l'amministratore del sistema può cambiare la configurazione.

In questa modalità detta "user networking", dalla macchina *guest* è possibile stabilire comunicazioni TCP e UDP verso l'esterno sfruttando lo stack di rete della macchina *host*. Attenzione: il traffico ICMP da e verso l'esterno viene scartato da Qemu.

I parametri della rete Qemu sono applicati automaticamente tramite DHCP (vedi `/etc/udhcpc/default.script`).

1.2 Macchine virtuali User Mode Linux

Dopo aver creato uno switch virtuale vde2, vi si possono collegare macchine virtuali User Mode Linux (UML)

```
1 # crea lo switch virtuale controllato con le pipe in /tmp/mioswitch
2 vde_switch -s /tmp/mioswitch
3 # per comodità usiamo un'altra shell
```

```
# l'interfaccia di rete eth0 è di tipo vde
# ed è collegata allo switch virtuale
# l'utenza uml è root senza password
/usr/local/share/uml/startBB.uml.txt eth0=vde,/tmp/mioswitch
# per comodità usiamo un'altra shell
/usr/local/share/uml/startBB.uml.txt eth0=vde,/tmp/mioswitch
```

In questa maniera abbiamo una rete locale virtuale fra le due macchine UML, che possiamo configurare a piacimento.

```
# Sulla prima
ip addr 172.16.0.1/24 dev eth0
ip link set eth0 up
# Sulla seconda
ip addr 172.16.0.2/24 dev eth0
ip link set eth0 up
# Sulla prima
nc -l -p 1234
# Sulla seconda
echo ciao | nc 172.16.0.1 1234
```

Questa rete consiste in uno stack interamente realizzato in user mode e pertanto del tutto sconosciuto al kernel del sistema Linux avviato da Qemu (come è facile controllare esaminando le interfacce con `ip addr`).

1.3 Rete virtuale collegata al kernel

Vde2 può anche essere usato per creare una rete virtuale *nota* al kernel che la ospita, grazie all'interfaccia apposita TUN/TAP che Linux mette a disposizione proprio a questi scopi. Per comodità le operazioni necessarie per attivare questa rete sono raccolte nello script `/etc/init.d/vde_switch`.

```
sudo /etc/init.d/vde_switch start
```

A questo punto l'interfaccia `tap0` risulterà connessa alla 192.168.240 (virtuale, controllata da `/run/vde.ctl`).

Strumenti di packet filtering

2.1 IPTables/Netfilter

Linux fornisce un'infrastruttura di *packet filtering* a livello kernel

- Linux 2.2 `ipchains` solo stateless filtering
- Linux ≥2.4 `netfilter` (kernel mode) permette di scrivere regole stateful, organizzate in *catene* e *tabelle* da `iptables` (user mode)

Le funzionalità di filtraggio sono estendibili con moduli kernel. Potete vedere quali moduli sono installati esplorando le directory del kernel in uso. Nella macchina virtuale:

```
1 ls /lib/modules/*/kernel/net/netfilter
```

Per avere un'idea di cosa fa un certo modulo si usa `modinfo` (per informazioni piú dettagliate sarà necessario consultare la documentazione specifica, però)

```
1 modinfo xt_mac
```

Il kernel di Linux mantiene alcune tabelle di regole da esaminare nella manipolazione dei pacchetti di rete. Il numero e la natura delle tabelle dipende dalla configurazione del kernel. Ogni tabella contiene *catene* di regole: con `iptables` si possono alterare le regole e aggiungere/togliere catene.

filter Tabella di default. Catene di regole: INPUT, FORWARD, OUTPUT;

nat Tabella consultata quando si *crea* una nuova connessione. PREROUTING, OUTPUT, POSTROUTING

mangle Tabella usata in manipolazioni particolari. PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING

raw Tabella ad alta priorità. PREROUTING, OUTPUT

Per vedere il contenuto di filter:

```
1 iptables -t filter -L
```

Per seguire il destino di un pacchetto consideriamo la tabella filter, rappresentata schematicamente in Fig. 2.1:

1. Arriva un pacchetto (p.es. dalla scheda di rete)
2. Si decide dove deve essere consegnato (routing)
3. Se è locale, passa per la INPUT chain (ACCEPT/DROP)

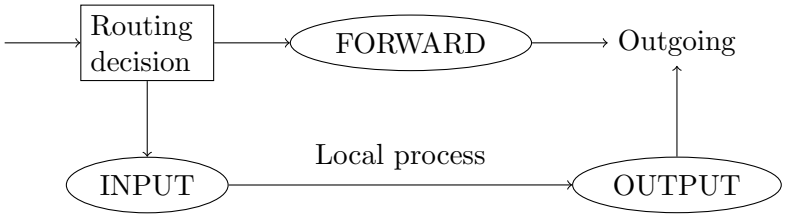

Figura 2.1: Tabella filter

4. Altrimenti DROP, ma se il forwarding è abilitato, si passa per la FORWARDING chain (ACCEPT/DROP)

5. Se un processo locale produce un pacchetto si passa per la OUTPUT chain (ACCEPT/DROP)

Il forwarding generalmente è disabilitato, a meno che non si tratti di un gateway con piú interfacce di rete.

Per scoprire se sulla macchina il forwarding è attivo

```
sysctl -a | grep forwarding
```

Nel caso lo si voglia attivare

```
sysctl -w net.ipv4.ip_forward=1
# equivalente a
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Questi parametri possono essere cambiati a run-time con `sysctl`, oppure resi permanenti (nel senso che vengono fissati al boot) mettendoli nel file `/etc/sysctl.conf`. A proposito: nel file `/etc/sysctl.conf` sono elencati altri parametri di configurazione del kernel che possono venire utili per difendere un nodo di una rete

```
# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1
```

Tornando a `iptables`, il grafo di flusso completo è schematizzato dalla Fig. 2.2.

La *routing decision* determina se il pacchetto ha destinazione locale o esterna: deve essere ripetuta piú volte perché il filtro potrebbe alterare i campi rilevanti del pacchetto.

2.1.1 Esercizi

- Leggere il manuale di `iptables`

- Interpretare l'effetto dei seguenti comandi

```
iptables -t filter -F # -t filter (default)
iptables -t filter -X
iptables -A INPUT --source 10.49.165.18 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -p tcp --dport 22 -j DROP
```

- Come si fa a salvare e ripristinare un insieme di regole?

*S*oluzione: prima cancella tutte le regole esistenti e poi blocca tutti gli accessi alla porta 22, meno quelli provenienti da 10.49.165.18. L'ultimo comando serve ad assicurare che il default sia DROP.

2.2 Rete fra host e guest

Qemu può essere configurato per fare NAT fra la rete host e guest. Per esempio se lo host è Windows, potrebbe contenere una chiamata a Qemu siffatta:

```
START qemu-system-i386w.exe -L Bios -vga std ^
-rtc base=localtime,clock=host ^
-cdrom ..\sicureti.iso ^
-net nic,model=e1000 -net user,hostfwd=tcp::8888-:80 ^
-no-acpi -no-hpet -no-reboot
```

L'opzione `hostfwd=tcp::8888:80` indica che le connessioni tcp alla porta 8888 dell'host (Windows) devono essere trasformate in connessioni alla porta 80 del guest (Linux). Su quale interfaccia arrivano? Senza ulteriori indicazioni sull'interfaccia "fisica" del guest, cioè `eth0` (10.0.2.15). Lo si può verificare attivando un web server sulla porta 10.0.2.15:80 e aprendo con il browser Windows l'url `http://localhost:8888`.

```
# sulla macchina guest
echo ciao > index.html
sudo httpd -f
```

Qualora sia attivata la rete TUN/TAP (vedi § 1.3), è attivo un web server sulla porta 192.168.240.6:80. Come fare per accedervi dal guest?

Occorre usare `iptables` per girare le connessioni a 10.0.2.15:80 a 192.168.240.6:80.

```
sudo -s
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.240.1:80
iptables -t nat -A POSTROUTING -j MASQUERADE
# necessario, altrimenti non c'è routing
echo 1 > /proc/sys/net/ipv4/ip_forward
```

(le credenziali del server web sono `admin:sicureti`)

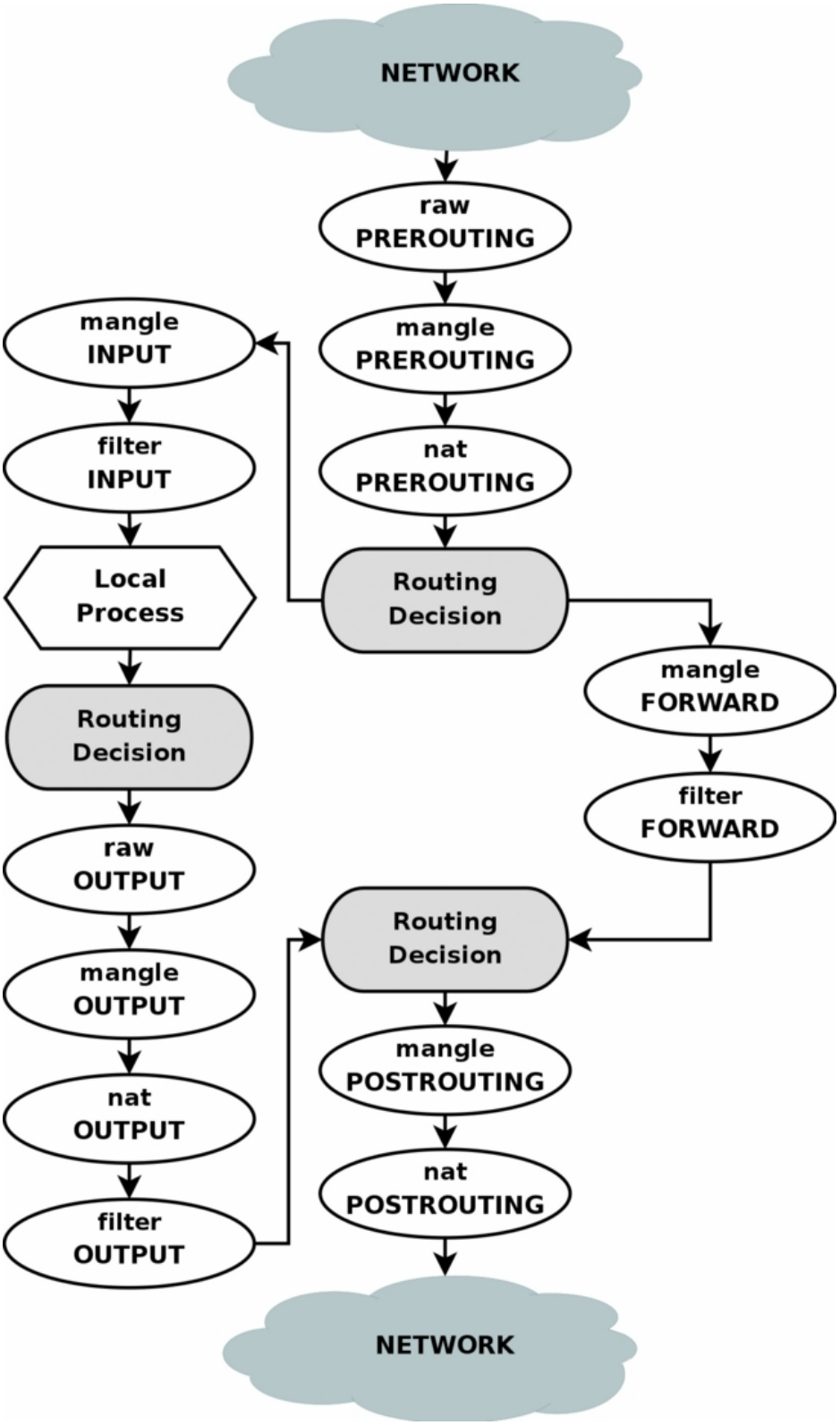

Figura 2.2: Flusso di un pacchetto fra le tabelle di `iptables`