Laboratorio di Reti e Sistemi Distribuiti

10: Interfacce virtuali

Roberto Marino, PhD[1]
roberto.marino@unime.it

[1]Dipartimento di Matematica, Informatica, Fisica e Scienze della Terra
Future Computing Research Laboratory
Università di Messina

Last Update: 20th March 2025

Interfacce virtuali

Cosa sono?

Le interfacce di rete virtuali in Linux sono dispositivi software che emulano le funzionalità delle interfacce fisiche, **ma non sono collegate a hardware reale**. Sono utilizzate per scopi come il testing, la creazione di VPN, il tunneling, o l'isolamento di reti.

Noi studieremo:

- TUN/TAP
- Veth Pair

Device file

In Linux, i dispositivi hardware e virtuali sono gestiti tramite device file nella directory `/dev`.

- `/dev/net/tun` è il device file ufficiale per il modulo del kernel TUN/TAP, che gestisce le interfacce di rete virtuali.
- Questo percorso è standardizzato e documentato nelle API del kernel, quindi tutte le applicazioni che usano TUN/TAP lo referenziano.

Interfacce TUN

Le interfacce TUN (da Network TUNnel) sono dispositivi virtuali **che operano a livello 3 dello stack OSI (strato di rete)**, gestendo pacchetti IP, non incapsulati in tramite ethernet. Sono utilizzate per:

1. Creare tunnel VPN (es. OpenVPN, WireGuard).
2. Iniettare/estrarre traffico IP da applicazioni userspace.
3. Simulare routing avanzato o scenari di rete complessi.

```
sudo ip tuntap add mode tun dev tun0
sudo ip addr add 10.0.0.1/24 dev tun0  # Assegna un IP
sudo ip link set tun0 up               # Attiva l'interfaccia
```

Le interfacce TUN non hanno un indirizzo MAC (lavorano a livello IP)

Usare le interfacce TUN

```c
#include <fcntl.h>
#include <linux/if_tun.h>
#include <string.h>
#include <sys/ioctl.h>
#include <unistd.h>

int main() {
    // Apri il dispositivo TUN
    int tun_fd = open("/dev/net/tun", O_RDWR);

    // Configura l'interfaccia
    struct ifreq ifr;
    memset(&ifr, 0, sizeof(ifr));
    ifr.ifr_flags = IFF_TUN | IFF_NO_PI;  // TUN (no Packet Information)
    strncpy(ifr.ifr_name, "tun0", IFNAMSIZ);

    // Crea l'interfaccia TUN
    ioctl(tun_fd, TUNSETIFF, &ifr);
```

Usare le interfacce TUN

```
    // Legge pacchetti IP dall'interfaccia
    char packet[1500];
    while (1) {
        int n = read(tun_fd, packet, sizeof(packet));
        printf("Pacchetto ricevuto (%d byte)\n", n);

        // Esempio: Invia una risposta (ping reply)
        // (ip\_header->daddr e ip\_header->saddr devono essere scambiati)
        // write(tun_fd, packet, n);
    }

    close(tun_fd);
    return 0;
}
```

Interfacce TAP

Le interfacce TAP (da Network TAP) sono dispositivi virtuali di rete **che operano a livello 2 dello stack OSI (strato di collegamento dati)**, gestendo frame Ethernet completi. Sono utilizzate per:

1. Collegare macchine virtuali (VM) o container a reti fisiche o virtuali.
2. Creare VPN di tipo bridged (a livello Ethernet).
3. Simulare dispositivi di rete in ambienti virtualizzati.

```
1 sudo ip tuntap add mode tap dev tap0    # Crea l'interfaccia
2 sudo ip link set tap0 up                # Attiva l'interfaccia
3 sudo ip addr add 192.168.100.1/24 dev tap0  # Assegna un IP (opz.)
```

A differenza delle interfacce TUN (che lavorano a livello IP), le TAP gestiscono frame Ethernet completi, inclusi header MAC e payload

Usare le interfacce TAP

```c
#include <fcntl.h>
#include <linux/if_tun.h>
#include <string.h>
#include <sys/ioctl.h>
#include <unistd.h>

int main() {
    int tap_fd = open("/dev/net/tun", O_RDWR);
    struct ifreq ifr;
    memset(&ifr, 0, sizeof(ifr));
    ifr.ifr_flags = IFF_TAP | IFF_NO_PI;  // Modalit  TAP
    strncpy(ifr.ifr_name, "tap0", IFNAMSIZ);

    // Crea l'interfaccia TAP
    ioctl(tap_fd, TUNSETIFF, &ifr);
```

Usare le interfacce TAP

```
    // Legge frame Ethernet
    char buffer[1500];
    while (1) {
        int n = read(tap_fd, buffer, sizeof(buffer));
        printf("Frame ricevuto (%d byte)\n", n);
    }

    close(tap_fd);
    return 0;
}
```

VETH Pair

Le veth sono interfacce virtuali Ethernet create sempre in coppia, come due estremità di un "cavo virtuale". Sono utilizzate per:

1. Collegare namespace di rete isolati (es. container Docker/Podman).
2. Connettere un namespace di rete a una rete bridge (es. in Kubernetes).
3. Simulare collegamenti punto-a-punto.

```
1 # Crea una coppia veth (veth0 <--> veth1)
2 sudo ip link add veth0 type veth peer name veth1
```

Il traffico inviato a veth0 viene ricevuto su veth1 (e viceversa).

Namespace di rete

I **network namespace** in Linux sono un meccanismo che permette di isolare le risorse di rete tra diversi processi o container. Ogni network namespace ha la propria istanza delle seguenti risorse di rete:

1. Interfacce di rete (eth0, lo, ecc.)
2. Tabella di routing
3. Regole di iptables
4. Stack TCP/IP separato
5. Connessioni di rete attive
6. Questo significa che un processo all'interno di un network namespace può avere la propria configurazione di rete indipendente dal namespace di rete principale (default).

Esempio di utilizzo VETH

Creare due namespace:

```
sudo ip netns add red
sudo ip netns add blue
```

Creare una coppia veth e collegala ai namespace:

```
sudo ip link add veth-red type veth peer name veth-blue
sudo ip link set veth-red netns red
sudo ip link set veth-blue netns blue
```

Assegnare IP e attivare le veth:

```
sudo ip netns exec red ip addr add 10.0.0.1/24 dev veth-red
sudo ip netns exec red ip link set veth-red up
sudo ip netns exec blue ip addr add 10.0.0.2/24 dev veth-blue
sudo ip netns exec blue ip link set veth-blue up
```

Esempi di utilizzo

1. QEMU usa interfacce **TAP** per connettere le VM a una rete fisica o virtuale.
2. OpenVPN incapsula pacchetti IP **dentro pacchetti IP** tramite **TUN**
3. Docker usa **VETH pair** per far dialogare container in namespace diversi