Esercitazione UDP

1. Uso di `ss` per mostrare le statistiche sull'uso delle socket

```
$ ss -s
Total: 243
TCP:   10 (estab 0, closed 0, orphaned 0, timewait 0)

Transport Total     IP        IPv6
RAW       0         0         0
UDP       4         3         1
TCP       10        3         7
INET      14        6         8
FRAG      0         0         0
```

Una **socket raw** consente l'accesso diretto al livello di rete e al livello di collegamento dati, bypassando lo stack TCP/IP standard e consentendo agli utenti di implementare protocolli al di sotto del livello di trasporto, monitorare le reti o sniffare i pacchetti.

La riga **INET** descrive tutte le socket Internet, pertanto è la somma delle righe per RAW, UDP e TCP.

La riga **FRAG** è relativa ai frammenti di pacchetti IP.

Informazioni analoghe sono esposte dal Kernel attraverso lo pseudo file system `proc` al path `/proc/net/sockstat`

2. Stampare tutte le socket UDP

```
$ ss -u
Recv-Q          Send-Q                    Local Address:Port
Peer Address:Port         Process
```

In questo caso non socket: infatti senza altre opzioni il comando `ss` mostra solo le socket "connesse", ovvero socket sui è stato associato sia un indirizzo locale sia un indirizzo remoto. Le socket UDP, in particolare quelle usate dai server, sono non connesse.

Possiamo usare l'opzione -a per mostrare le socket in tutti gli stati (si noti che nel caso del comando `ss` e di molti altri comandi le opzioni brevi possono essere scritte separate `-u -a` oppure combinate `-ua`).

```
$ ss -ua
State        Recv-Q       Send-Q                Local
Address:Port                  Peer Address:Port       Process
UNCONN       0            0
127.0.0.53%lo:domain                   0.0.0.0:*
UNCONN       0            0
10.255.255.254:domain                   0.0.0.0:*
UNCONN       0            0
127.0.0.1:323                         0.0.0.0:*
UNCONN       0            0
[::1]:323                            [::]:*
```

Le colonne forniscono:

- Lo stato della socket
- Il numero di byte ricevuti in attesa di essere consegnati al processo
- Il numero di byte in attesa di essere trasmessi o, se trasmessi, in attesa di riscontro
- L'indirizzo locale
- L'indirizzo remoto

Avremmo potuto usare l'opzione `-l` in alternativa alla opzione `-a`: nel caso delle socket UDP, seleziona le socket connesse (nel caso di TCP, seleziona invece le socket in attesa di richieste di connessione).

3. Usando l'opzione `-n` evitiamo la traduzione del numero di porta nel nome del servizio (sulla base del file `/etc/services`) e `-p` stampiamo le informazioni sul processo. Per vedere le informazioni sui processi potrebbe essere necessario eseguire il comando con `sudo`.

4. L'opzione `-E` ci mostra le socket **quando sono distrutte** (può essere terminato con `CTRL-C`)

   ```
   $ ss -uanE
   ```

5. In un altro terminale, eseguiamo `wireshark` e iniziamo la cattura su `eth0`

6. In un altro terminale, usiamo il comando dig e vediamo che viene stampato qualcosa da `ss`

   ```
   $ dig @dns.uniroma2.it uniroma2.it MX
   ```

   Dovremmo trovare una riga per una socket UDP, che è stata associata all'indirizzo remoto del resolver DNS, attualmente `160.80.1.3` e numero di porta `53` (questo

perché `dig` ha invocato la chiamata di sistema `connect` per specificare la parte remota della propria socket UDP)

7. In `wireshark` possiamo usare il filtro `dns` per trovare i pacchetti DNS e troviamo la richiesta

8. Analizziamo le intestazioni

    a. IP:
        i. indirizzo IP sorgente e destinazione (devono corrispondere all'output di `ss` e ip addr)

```
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc
noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd
00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet 10.255.255.254/32 brd 10.255.255.254 scope
global lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
qdisc mq state UP group default qlen 1000
    link/ether 00:15:5d:9f:62:e3 brd
ff:ff:ff:ff:ff:ff
    inet 172.21.242.109/20 brd 172.21.255.255 scope
global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::215:5dff:fe9f:62e3/64 scope link
       valid_lft forever preferred_lft forever
```

        ii. Protocol: UDP (dice che il payload va passato a UDP invece che a TCP, per esempio)
    b. UDP
        i. Numero di porta sorgente e destinazione (devono corrispondere all'output di `ss`)
        ii. Checksum
        iii. Length: mostrare che è 8 + lunghezza del pacchetto DNS incapsulato
9. Rivedere brevemente DNS: far riferimento alle slide e alla esercitazione

Osservazioni sui filtri Wireshark

- Facendo tasto destro su qualunque campo di può importare quel campo + valore come filtro
- Esempi di filtri:
  - `ip. dst == 160.80.1.3` (o `eq`)
  - `ip. dst != 160.80.1.3` (o `neq`)
  - per gli indirizzi IP possono indicare un prefisso di rete in notazione CIDR: `ip.src == 172.21.240.0/20`
  - la stessa cosa si può fare con `ip.dst`
  - `udp.dstport == 53` (o `eq`)
  - `udp.dstport != 53` (o `ne`)
  - la stessa cosa si può fare con `udp.srcport`
  - `udp.dstport in {53, 54, 55}` oppure `udp.dstport in {53..55}` oppure `udp.dstport in {53..54, 55}`
  - support I connettivi logici `and` (`&&`), `or` (`||`), `not` (`!`), `xor` (`^^`)
  - possono anche usare operatori come < (`lt`), > (`gt`), <= (`le`), >= (`ge`)

Riferimenti Utili:

- https://manpages.ubuntu.com/manpages/jammy/man8/ss.8.html
- https://www.wireshark.org/docs/man-pages/wireshark-filter.html