**Materiale addizionale su ss**

*Queste note si aggiungono, ma non si sostituiscono, a quanto scritto nel word della lezione precedente.*

Riferimento alla man page: <https://manpages.ubuntu.com/manpages/jammy/en/man8/ss.8.html>

Significato delle colonne nell’output di ss:

* **Netid** – Type of socket. Common types are *TCP*, *UDP*, *u\_str* (Unix stream), and *u\_seq* (Unix sequence).
* **State** – State of the socket. Most commonly *ESTAB* (established), *UNCONN* (unconnected), *LISTEN* (listening).
* **Recv-Q** – Number of received packets in the queue.
* **Send-Q** – Number of sent packets in the queue.
* **Local address:port** – Address of local machine and port.
* **Peer address:port** – Address of remote machine and port.

Le opzioni possono essere scritte separatamente oppure combinate:

ss -t -u

è la stessa cosa che

ss -tu

Si noti che le due opzioni vanno interpretate in “or”, nel senso che saranno restituite socket relative al protocollo TCP oppure UDP.

Altre opzioni interessanti (prese da)

**-o, --options**

Show timer information. For TCP protocol, the output

format is:

timer:(<timer\_name>,<expire\_time>,<retrans>)

**<timer\_name>**

the name of the timer, there are five kind of timer

names:

**on** : means one of these timers: TCP retrans timer,

TCP early retrans timer and tail loss probe timer

**keepalive**: tcp keep alive timer

**timewait**: timewait stage timer

**persist**: zero window probe timer

**unknown**: none of the above timers

**<expire\_time>**

how long time the timer will expire

**<retrans>**

how many times the retransmission occurred

**-m, --memory**

Show socket memory usage. The output format is:

skmem:(r<rmem\_alloc>,rb<rcv\_buf>,t<wmem\_alloc>,tb<snd\_buf>,

f<fwd\_alloc>,w<wmem\_queued>,o<opt\_mem>,

bl<back\_log>,d<sock\_drop>)

**<rmem\_alloc>**

the memory allocated for receiving packet

**<rcv\_buf>**

the total memory can be allocated for receiving

packet

**<wmem\_alloc>**

the memory used for sending packet (which has been

sent to layer 3)

**<snd\_buf>**

the total memory can be allocated for sending

packet

**<fwd\_alloc>**

the memory allocated by the socket as cache, but

not used for receiving/sending packet yet. If need

memory to send/receive packet, the memory in this

cache will be used before allocate additional

memory.

**<wmem\_queued>**

The memory allocated for sending packet (which has

not been sent to layer 3)

**<opt\_mem>**

The memory used for storing socket option, e.g.,

the key for TCP MD5 signature

**<back\_log>**

The memory used for the sk backlog queue. On a

process context, if the process is receiving

packet, and a new packet is received, it will be

put into the sk backlog queue, so it can be

received by the process immediately

**<sock\_drop>**

the number of packets dropped before they are de-

multiplexed into the socket

**-i,** **--info**

Show internal TCP information. Below fields may appear:

**ts** show string "ts" if the timestamp option is set

**sack** show string "sack" if the sack option is set

**ecn** show string "ecn" if the explicit congestion notification option is set

**ecnseen**

show string "ecnseen" if the saw ecn flag is found in received packets

**fastopen**

show string "fastopen" if the fastopen option is set

**cong\_alg**

the congestion algorithm name, the default congestion algorithm is "cubic"

**wscale:<snd\_wscale>:<rcv\_wscale>**

if window scale option is used, this field shows the send scale factor and

receive scale factor

**rto:<icsk\_rto>**

tcp re-transmission timeout value, the unit is millisecond

**backoff:<icsk\_backoff>**

used for exponential backoff re-transmission, the actual re-transmission

timeout value is icsk\_rto << icsk\_backoff

**rtt:<rtt>/<rttvar>**

rtt is the average round trip time, rttvar is the mean deviation of rtt,

their units are millisecond

**ato:<ato>**

ack timeout, unit is millisecond, used for delay ack mode

**mss:<mss>**

max segment size

**cwnd:<cwnd>**

congestion window size

**pmtu:<pmtu>**

path MTU value

**ssthresh:<ssthresh>**

tcp congestion window slow start threshold

**bytes\_acked:<bytes\_acked>**

bytes acked

**bytes\_received:<bytes\_received>**

bytes received

**segs\_out:<segs\_out>**

segments sent out

**segs\_in:<segs\_in>**

segments received

**send** **<send\_bps>bps**

egress bps

**lastsnd:<lastsnd>**

how long time since the last packet sent, the unit is millisecond

**lastrcv:<lastrcv>**

how long time since the last packet received, the unit is millisecond

**lastack:<lastack>**

how long time since the last ack received, the unit is millisecond

**pacing\_rate** **<pacing\_rate>bps/<max\_pacing\_rate>bps**

the pacing rate and max pacing rate

**rcv\_space:<rcv\_space>**

a helper variable for TCP internal auto tuning socket receive buffer

**tcp-ulp-mptcp** **flags:[MmBbJjecv]** **token:<rem\_token(rem\_id)/loc\_token(loc\_id)>**

**seq:<sn>** **sfseq:<ssn>** **ssnoff:<off>** **maplen:<maplen>**

MPTCP subflow information

La sintassi del comando ss è:

**ss** [options] [ FILTER ]

Abbiamo già visto alcune opzioni, cosa possiamo dire del filter?

Si compone di due parti opzionali:

**[** **state** **STATE-FILTER** **]** **[** **EXPRESSION** **]**

**STATE-FILTER** permette di filtrare le socket usando una sequenza di identificatori di stato e esclusioni.

Possiamo usare tutti gli stati TCP:

**established**, **syn-sent**, **syn-recv**, **fin-wait-1**, **fin-wait-2**, **time-wait**, **closed**,

**close-wait**, **last-ack**, **listening**, **closing**

Cui si aggiungono questi identificatori:

**all** – per tutti gli stati

**connected** – tutti gli stati tranne **listening** e **closed**

**synchronized** – tutti gli stati **connected** tranne che **syn-sent**

**bucket** – stati che sono mantenuti come minisocket, es. **time-wait** e **syn-recv**

**big** – opposto di **bucket**

**EXPRESSION** permette di combinare diversi predicati usando operatori logici and (o && o &), or (o || o |), not (o !). Se non viene indicato un operatore si assume and.

Alcuni predicati:

**{dst|src}** **[=]** **HOST**

Testa se la sorgente o la destinazione corrispondono all’HOST (vedi sotto).

**{dport|sport}** **[OP]** **[FAMILY:]:PORT**

Confronta la porta di destinazione o sorgente con porta. OP può essere uno tra "<", "<=", "=","!=", ">=" and ">".

**autobound**

Testa se la porta (o il path nel caso di altri tipi di socket diverse da quelle Internet) è stata allocata dinamicamente.

HOST può essere definito come un hostname DNS o indirizzo IP secondo le convenzioni di IPv4 (dot-decimal notation) o IPv6

PORT può essere dato come numero oppure usando uno dei nomi che sono listati in /etc/services (che sono gli stessi nomi usati senza l’opzione -n).

Sia PORT sia HOST possono essere sostituiti da un asterisco (come *wildcard*).

Esempio (notate lo spazio tra le parentesi):

ss -t state established '( dport == :http and dst == 160/8 and autobound )'