SS Utility: Quick Intro

Alexey Kuznetosv, kuznet@ms2.inr.ac.ru some_negative_number, 20 Sep 2001

ss is one another utility to investigate sockets. Functionally it is NOT better than netstat combined with some perl/awk scripts and though it is surely faster it is not enough to make it much better. :-) So, stop reading this now and do not waste your time. Well, certainly, it proposes some functionality, which current netstat is still not able to do, but surely will soon.

1 Why?

/proc interface is inadequate, unfortunately. When amount of sockets is enough large, netstat or even plain cat /proc/net/tcp/ cause nothing but pains and curses. In linux-2.4 the desease became worse: even if amount of sockets is small reading /proc/net/tcp/ is slow enough.

This utility presents a new approach, which is supposed to scale well. I am not going to describe technical details here and will concentrate on description of the command. The only important thing to say is that it is not so bad idea to load module tcp_diag, which can be found in directory Modules of iproute2. If you do not make this ss will work, but it falls back to /proc and becomes slow like netstat, well, a bit faster yet (see section "Some numbers").

2 Old news

In the simplest form ss is equivalent to netstat with some small deviations.

- ss -t -a dumps all TCP sockets
- ss -u -a dumps all UDP sockets
- ss -w -a dumps all RAW sockets
- ss -x -a dumps all UNIX sockets

Option -o shows TCP timers state. Option -e shows some extended information. Etc. etc. etc. Seems, all the options of netstat related to sockets are supported. Though not AX.25 and other bizarres. :-) If someone wants, he can make support for decnet and ipx. Some rudimentary support for them is already present in iproute2 libutils, and I will be glad to see these new members.

However, standard functionality is a bit different:

The first: without option -a sockets in states TIME-WAIT and SYN-RECV are skipped too. It is more reasonable default, I think.

The second: format of UNIX sockets is different. It coincides with tcp/udp. Though standard kernel still does not allow to see write/read queues and peer address of connected UNIX sockets, the patch doing this exists.

The third: default is to dump only TCP sockets, rather than all of the types.

The next: by default it does not resolve numeric host addresses (like `ip`)! Resolving is enabled with option `-r`. Service names, usually stored in local files, are resolved by default. Also, if service database does not contain references to a port, `ss` queries system `rpcbind`. RPC services are prefixed with `rpc.` Resolution of services may be suppressed with option `-n`.

It does not accept "long" options (I dislike them, sorry). So, address family is given with family identifier following option `-f` to be algined to iproute2 conventions. Mostly, it is to allow option parser to parse addresses correctly, but as side effect it really limits dumping to sockets supporting only given family. Option `-A` followed by list of socket tables to dump is also supported. Logically, id of socket table is different of _address_ family, which is another point of incompatibility. So, id is one of `all`, `tcp`, `udp`, `raw`, `inet`, `unix`, `packet`, `netlink`. See? Well, `inet` is just abbreviation for `tcp|udp|raw` and it is not difficult to guess that `packet` allows to look at packet sockets. Actually, there are also some other abbreviations, f.e. `unix_dgram` selects only datagram UNIX sockets.

The next: well, I still do not know. :-)

3 Time to talk about new functionality.

It is builtin filtering of socket lists.

3.1 Filtering by state.

`ss` allows to filter socket states, using keywords `state` and `exclude`, followed by some state identifier.

State identifier are standard TCP state names (not listed, they are useless for you if you already do not know them) or abbreviations:

- `all` - for all the states
- `bucket` - for TCP minisockets (`TIME-WAIT|SYN-RECV`)
- `big` - all except for minisockets
- `connected` - not closed and not listening
- `synchronized` - connected and not `SYN-SENT`

F.e. to dump all tcp sockets except `SYN-RECV`:

```
ss exclude SYN-RECV
```

If neither `state` nor `exclude` directives are present, state filter defaults to `all` with option `-a` or to `all`, excluding listening, syn-recv, time-wait and closed sockets.

3.2 Filtering by addresses and ports.

Option list may contain address/port filter. It is boolean expression which consists of boolean operation `or`, `and`, `not` and predicates. Actually, all the flavors of names for boolean operations are eaten: `&`, `&&`, |, ||, `!`, but do not forget about special sense given to these symbols by unix shells and escape them correctly, when used from command line.

Predicates may be of the folowing kinds:

- A. Address/port match, where address is checked against mask and port is either wildcard or exact. It is one of:

  ```
  dst prefix:port
  src prefix:port
  src unix:STRING
  src link:protocol:ifindex
  src nl:channel:pid
  ```

  Both prefix and port may be absent or replaced with `*`, which means wildcard. UNIX socket use more powerful scheme matching to socket names by shell wildcards. Also, prefixes unix: and link: may be omitted, if address family is evident from context (with option `-x` or with `-f unix` or with `unix` keyword)

  F.e.

  ```
  dst 10.0.0.1
  dst 10.0.0.1:
  dst 10.0.0.1/32:
  dst 10.0.0.1:*
  ```

  are equivalent and mean socket connected to any port on host 10.0.0.1

  ```
  dst 10.0.0.0/24:22
  ```

  sockets connected to port 22 on network 10.0.0.0...255.

  Note that port separated of address with colon, which creates troubles with IPv6 addresses. Generally, we interpret the last colon as splitting port. To allow to give IPv6 addresses, trick like used in IPv6 HTTP URLs may be used:

  ```
  dst [::1]
  ```

  are sockets connected to ::1 on any port

  Another way is `dst ::1`128/. / helps to understand that colon is part of IPv6 address.

  Now we can add another alias for `dst 10.0.0.1`: `dst [10.0.0.1]`. :-)

  Address may be a DNS name. In this case all the addresses are looked up (in all the address families, if it is not limited by option `-f` or special address prefix `inet:`, `inet6`) and resulting expression is `or` over all of them.

- B. Port expressions:

  ```
  dport >= :1024
  dport != :22
  sport < :32000
  ```

etc.

All the relations: <, >, =, >=, =, ==, `!=`, `eq`, `ge`, `lt`, `ne`... Use variant which you like more, but not forget to escape special characters when typing them in command line. :-)

Note that port number syntactically coincides to the case A! You may even add an IP address, but it will not participate incomparison, except for == and `!=`, which are equivalent to corresponding predicates of type A. F.e. `dst 10.0.0.1:22` is equivalent to `dport eq 10.0.0.1:22` and `not dst 10.0.0.1:22` is equivalent to `dport neq 10.0.0.1:22`

- C. Keyword `autobound`. It matches to sockets bound automatically on local system.

4 Examples

- 1. List all the tcp sockets in state `FIN-WAIT-1` for our apache to network 193.233.7/24 and look at their timers:

  ```
  ss -o state fin-wait-1 \( sport = :http or sport = :https \) \
                        dst 193.233.7/24
  ```

  Oops, forgot to say that missing logical operation is equivalent to `and`.

- 2. Well, now look at the rest...

  ```
  ss -o excl fin-wait-1
  ss state fin-wait-1 \( sport neq :http and sport neq :https \) \
                     or not dst 193.233.7/24
  ```

  Note that we have to do _two_ calls of ss to do this. State match is always anded to address/port match. The reason for this is purely technical: ss does fast skip of not matching states before parsing addresses and I consider the ability to skip fastly gobs of time-wait and syn-recv sockets as more important than logical generality.

- 3. So, let's look at all our sockets using autobound ports:

  ```
  ss -a -A all autobound
  ```

- 4. And eventually find all the local processes connected to local X servers:

  ```
  ss -xp dst "/tmp/.X11-unix/*"
  ```

  Pardon, this does not work with current kernel, patching is required. But we still can look at server side:

  ```
  ss -x src "/tmp/.X11-unix/*"
  ```

5 Returning to ground: real manual

5.1 Command arguments

General format of arguments to `ss` is:

```
ss [ OPTIONS ] [ STATE-FILTER ] [ ADDRESS-FILTER ]
```

5.1.1 OPTIONS

`OPTIONS` is list of single letter options, using common unix conventions.

- `-h` - show help page
- `-?` - the same, of course
- `-v`, `-V` - print version of `ss` and exit
- `-s` - print summary statistics. This option does not parse socket lists obtaining summary from various sources. It is useful when amount of sockets is so huge that parsing `/proc/net/tcp` is painful.
- `-D FILE` - do not display anything, just dump raw information about TCP sockets to `FILE` after applying filters. If `FILE` is `-` `stdout` is used.
- `-F FILE` - read continuation of filter from `FILE`. Each line of `FILE` is interpreted like single command line option. If `FILE` is `-` `stdin` is used.
- `-r` - try to resolve numeric address/ports
- `-n` - do not try to resolve ports
- `-o` - show some optional information, f.e. TCP timers
- `-i` - show some infomration specific to TCP (RTO, congestion window, slow start threshould etc.)
- `-e` - show even more optional information
- `-m` - show extended information on memory used by the socket. It is available only with `tcp_diag` enabled.
- `-p` - show list of processes owning the socket
- `-f FAMILY` - default address family used for parsing addresses. Also this option limits listing to sockets supporting given address family. Currently the following families are supported: `unix`, `inet`, `inet6`, `link`, `netlink`.
- `-4` - alias for `-f inet`
- `-6` - alias for `-f inet6`
- `-0` - alias for `-f link`
- `-A LIST-OF-TABLES` - list of socket tables to dump, separated by commas. The following identifiers are understood: `all`, `inet`, `tcp`, `udp`, `raw`, `unix`, `packet`, `netlink`, `unix_dgram`, `unix_stream`, `packet_raw`, `packet_dgram`.
- `-x` - alias for `-A unix`
- `-t` - alias for `-A tcp`
- `-u` - alias for `-A udp`
- `-w` - alias for `-A raw`
- `-a` - show sockets of all the states. By default sockets in states `LISTEN`, `TIME-WAIT`, `SYN_RECV` and `CLOSE` are skipped.
- `-l` - show only sockets in state `LISTEN`

5.1.2 STATE-FILTER

`STATE-FILTER` allows to construct arbitrary set of states to match. Its syntax is sequence of keywords `state` and `exclude` followed by identifier of state. Available identifiers are:

- All standard TCP states: `established`, `syn-sent`, `syn-recv`, `fin-wait-1`, `fin-wait-2`, `time-wait`, `closed`, `close-wait`, `last-ack`, `listen` and `closing`.
- `all` - for all the states
- `connected` - all the states except for `listen` and `closed`
- `synchronized` - all the `connected` states except for `syn-sent`
- `bucket` - states, which are maintained as minisockets, i.e. `time-wait` and `syn-recv`.
- `big` - opposite to `bucket`

5.1.3 ADDRESS_FILTER

`ADDRESS_FILTER` is boolean expression with operations `and`, `or` and `not`, which can be abbreviated in C style f.e. as `&`, `&&`.

Predicates check socket addresses, both local and remote. There are the following kinds of predicates:

- `dst ADDRESS_PATTERN` - matches remote address and port
- `src ADDRESS_PATTERN` - matches local address and port
- `dport RELOP PORT` - compares remote port to a number
- `sport RELOP PORT` - compares local port to a number
- `autobound` - checks that socket is bound to an ephemeral port

`RELOP` is some of <=, >=, == etc. To make this more convinient for use in unix shell, alphabetic FORTRAN-like notations `le`, `gt` etc. are accepted as well.

The format and semantics of `ADDRESS_PATTERN` depends on address family.

- `inet` - `ADDRESS_PATTERN` consists of IP prefix, optionally followed by colon and port. If prefix or port part is absent or replaced with `*`, this means wildcard match.
- `inet6` - The same as `inet`, only prefix refers to an IPv6 address. Unlike `inet` colon becomes ambiguous, so that `ss` allows to use scheme, like used in URLs, where address is suppounded with `[ ... ]`.
- `unix` - `ADDRESS_PATTERN` is shell-style wildcard.
- `packet` - format looks like `inet`, only interface index stays instead of port and link layer protocol id instead of address.
- `netlink` - format looks like `inet`, only socket pid stays instead of port and netlink channel instead of address.

`PORT` is syntactically `ADDRESS_PATTERN` with wildcard address part. Certainly, it is undefined for UNIX sockets.

5.2 Environment variables

`ss` allows to change source of information using various environment variables:

- `PROC_SLABINFO` to override `/proc/slabinfo`
- `PROC_NET_TCP` to override `/proc/net/tcp`
- `PROC_NET_UDP` to override `/proc/net/udp`
- etc.

Variable `PROC_ROOT` allows to change root of all the `/proc/` hierarchy.

Variable `TCPDIAG_FILE` prescribes to open a file instead of requesting kernel to dump information about TCP sockets.

This option is used mainly to investigate bug reports, when dumps of files usually found in `/proc/` are recevied by e-mail.

5.3 Output format

Six columns. The first is `Netid`, it denotes socket type and transport protocol, when it is ambiguous: `tcp`, `udp`, `raw`, `u_str` is abbreviation for `unix_stream`, `u_dgr` for UNIX datagram sockets, `nl` for netlink, `p_raw` and `p_dgr` for raw and datagram packet sockets. This column is optional, it will be hidden, if filter selects an unique netid.

The second column is `State`. Socket state is displayed here. The names are standard TCP names, except for `UNCONN`, which cannot happen for TCP, but normal for not connected sockets of another types. Again, this column can be hidden.

Then two columns (`Recv-Q` and `Send-Q`) showing amount of data queued for receive and transmit.

And the last two columns display local address and port of the socket and its peer address, if the socket is connected.

If options `-o`, `-e` or `-p` were given, options are displayed not in fixed positions but separated by spaces pairs: `option:value`. If value is not a single number, it is presented as list of values, enclosed to `( ... )` and separated with commas. F.e.

```
timer:(keepalive,111min,0)
```

is typical format for TCP timer (option `-o`).

```
users:((X,113,3))
```

is typical for list of users (option `-p`).

6 Some numbers

Well, let us use `pidentd` and a tool `ibench` to measure its performance. It is 30 requests per second here. Nothing to test, it is too slow. OK, let us patch pidentd with patch from directory Patches. After this it handles about 4300 requests per second and becomes handy tool to pollute socket tables with lots of timewait buckets.

So, each test starts from pollution tables with 30000 sockets and then doing full dump of the table piped to wc and measuring timings with time:

Results:

- `netstat -at` - 15.6 seconds
- `ss -atr`, but without `tcp_diag` - 5.4 seconds
- `ss -atr` with `tcp_diag` - 0.47 seconds

No comments. Though one comment is necessary, most of time without `tcp_diag` is wasted inside kernel with completely blocked networking. More than 10 seconds, yes. `tcp_diag` does the same work for 100 milliseconds of system time.