
tcconfig Documentation

Release 0.21.3

Tsuyoshi Hombashi

Jan 05, 2019

Table of Contents

1	tcconfig	1
1.1	Summary	1
1.2	Traffic control	1
1.2.1	Setup traffic shaping rules	1
1.2.2	Available parameters	1
1.2.3	Targets	2
2	Installation	3
2.1	Install via pip (recommended)	3
2.2	Install in Debian/Ubuntu from a deb package	3
3	Dependencies	5
3.1	Linux packages	5
3.2	Linux kernel module	5
3.3	Python packages	5
3.3.1	Optional Python packages	6
3.3.2	Test dependencies	6
4	Usage	7
4.1	Set traffic control (<code>tcset</code> command)	7
4.1.1	<code>tcset</code> command help	7
4.1.2	Basic usage	9
4.1.2.1	e.g. Set a limit on bandwidth up to 100Kbps	9
4.1.2.2	e.g. Set network latency	10
4.1.2.3	e.g. Set 0.1% packet loss	10
4.1.2.4	e.g. All of the above settings at once	10
4.1.2.5	e.g. Specify the IP address of traffic control	10
4.1.2.6	e.g. Specify the IP network and port of traffic control	10
4.1.3	Advanced usage	11
4.1.3.1	Traffic control of incoming packets	11
4.1.3.2	Set latency distribution	11
4.1.3.3	Set multiple traffic shaping rules to an interface	11
4.1.3.4	Using IPv6	11
4.1.3.5	Get <code>tc</code> commands	12
4.1.3.6	Generate a <code>tc</code> script file	12
4.1.3.7	Set a shaping rule for multiple destinations	12
4.1.3.8	Shaping rules for between multiple hosts	13

4.1.4	Docker container	13
4.1.4.1	Set traffic control to a docker container	13
4.1.4.2	Set traffic control within a docker container	14
4.2	Delete traffic control (<code>tc del</code> command)	14
4.2.1	<code>tc del</code> command help	14
4.2.1.1	e.g. Delete traffic control of <code>eth0</code>	15
4.2.2	Advanced usage	15
4.3	Display traffic control configurations (<code>tc show</code> command)	16
4.3.1	<code>tc show</code> command help	17
4.3.1.1	Example	17
4.4	Backup and restore traffic control configurations	18
4.4.1	e.g. Backup configurations	18
4.4.2	e.g. Restore configurations	18
4.5	Execute with not super-user	19
5	Troubleshooting	21
5.1	RTNETLINK answers: No such file or directory	21
5.1.1	Phenomenon	21
5.1.2	Solution	21
5.2	RTNETLINK answers: Operation not supported	22
5.2.1	Phenomenon	22
5.2.2	Solutions	22
6	Indices and tables	25
7	Links	27
8	Indices and tables	29

1.1 Summary

A tc command wrapper. Make it easy to set up traffic control of network bandwidth/latency/packet-loss/packet-corruption/etc. to a network-interface/Docker-container(veth).

1.2 Traffic control

1.2.1 Setup traffic shaping rules

Easy to apply traffic shaping rules to specific network:

- Outgoing/Incoming packets
- Source/Destination IP-address/network (IPv4/IPv6)
- Source/Destination ports

1.2.2 Available parameters

The following parameters can be set to network interfaces:

- Network bandwidth rate [G/M/K bps]
- Network latency [microseconds/milliseconds/seconds/minutes]
- Packet loss rate [%]
- Packet corruption rate [%]
- Packet duplicate rate [%]

- Packet reordering rate [%]

1.2.3 Targets

- Network interfaces: e.g. `eth0`
- Docker container (`veth` corresponding with a container)

2.1 Install via pip (recommended)

tcconfig can be installed from PyPI via pip (Python package manager) command.

```
sudo pip install tcconfig
```

2.2 Install in Debian/Ubuntu from a deb package

1. `wget https://github.com/thombashi/tcconfig/releases/download/<version>/tcconfig-<version>-amd64.deb`
2. `dpkg -iv tcconfig-<version>-amd64.deb`

Example

```
$ wget https://github.com/thombashi/tcconfig/releases/download/v0.19.0/  
↪tcconfig_0.19.0_amd64.deb  
$ sudo dpkg -i tcconfig_0.19.0_amd64.deb
```


Python 2.7+ or 3.4+

3.1 Linux packages

- **mandatory: required for `tc` command:**
 - *Ubuntu/Debian*: `iproute2`
 - *Fedora/RHEL*: `iproute-tc`
- **optional: required to when you use `--iptables` option:**
 - `iptables`

3.2 Linux kernel module

- `sch_netem`

3.3 Python packages

Dependency python packages are automatically installed during `tcconfig` installation via `pip`.

- `DataProperty`
- `ipaddress`
- `logbook`
- `msgfy`
- `pyparsing`

- six
- subprocessrunner
- typepy
- voluptuous

3.3.1 Optional Python packages

- **netifaces**
 - Suppress excessive error messages if this package installed
- Pygments

3.3.2 Test dependencies

- allpairspy
- pingparsing
- pytest
- pytest-runner
- tox

4.1 Set traffic control (`tcset` command)

`tcset` is a command to add traffic control rule to a network interface (device).

You can delete rule(s) from a network interface by *Delete traffic control* (`tcdel` command).

4.1.1 `tcset` command help

```
usage: tcset [-h] [-V] [--tc-command | --tc-script] [--debug | --quiet]
            [--stacktrace] [--import-setting]
            [--overwrite | --change | --add] [--rate BANDWIDTH_RATE]
            [--delay NETWORK_LATENCY] [--delay-distro LATENCY_DISTRO_TIME]
            [--loss PACKET_LOSS_RATE] [--duplicate PACKET_DUPLICATE_RATE]
            [--corrupt CORRUPTION_RATE] [--reordering REORDERING_RATE]
            [--shaping-algo {htb,tbf}] [--iptables]
            [--direction {outgoing,incoming}] [--network DST_NETWORK]
            [--src-network SRC_NETWORK] [--port DST_PORT]
            [--src-port SRC_PORT] [--ipv6]
            [--exclude-dst-network EXCLUDE_DST_NETWORK]
            [--exclude-src-network EXCLUDE_SRC_NETWORK]
            [--exclude-dst-port EXCLUDE_DST_PORT]
            [--exclude-src-port EXCLUDE_SRC_PORT] [--docker]
            [--src-container SRC_CONTAINER] [--dst-container DST_CONTAINER]
            device

positional arguments:
  device                target name: network-interface/config-file (e.g. eth0)

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  --tc-command          display tc commands to be executed and exit. these
```

(continues on next page)

(continued from previous page)

	commands are not actually executed.
--tc-script	generate a shell script file that described tc commands. this tc script execution result nearly equivalent with the tcconfig command. the script can be executed without tcconfig package installation.
--debug	for debug print .
--quiet	suppress execution log messages.
--import-setting	import traffic control settings from a configuration file.
--overwrite	overwrite existing traffic shaping rules.
--change	change existing traffic shaping rules to the new one. this option is effective to reduce the time between the shaping rule switching compared to --overwrite option. note: just adds a shaping rule if there are no existing shaping rules.
--add	add a traffic shaping rule in addition to existing rules.
Debug:	
--stacktrace	print stack trace for debug information. --debug option required to see the debug print .
Traffic Control Parameters:	
--rate BANDWIDTH_RATE, --bandwidth-rate BANDWIDTH_RATE	network bandwidth rate [bit per second]. valid units are either: K/M/G/Kbps/Mbps/Gbps e.g. --rate 10Mbps
--delay NETWORK_LATENCY	round trip network delay. the valid range is from 0ms to 60min. valid time units are: m/min/mins/minute/minutes/s/sec/secs/second/seconds/ms/msec/msecs/millisecond/milliseconds/us/usec/usecs/microsecond/microseconds. if no unit string found, considered milliseconds as the time unit. (default=0ms)
--delay-distro LATENCY_DISTRO_TIME	distribution of network latency becomes X +- Y (normal distribution). Here X is the value of --delay option and Y is the value of --delay-dist option). network latency distribution is uniform, without this option. valid time units are: m/min/mins/minute/minutes/s/sec/secs/second/seconds/ms/msec/msecs/millisecond/milliseconds/us/usec/usecs/microsecond/microseconds. if no unit string found, considered milliseconds as the time unit.
--loss PACKET_LOSS_RATE	round trip packet loss rate [%]. the valid range is from 0 to 100. (default=0)
--duplicate PACKET_DUPLICATE_RATE	round trip packet duplicate rate [%]. the valid range is from 0 to 100. (default=0)
--corrupt CORRUPTION_RATE	packet corruption rate [%]. the valid range is from 0 to 100. packet corruption means single bit error at a random offset in the packet. (default=0)
--reordering REORDERING_RATE	packet reordering rate [%]. the valid range is from 0 to 100. (default=0)
--shaping-algo {htb,tbf}	

(continues on next page)

(continued from previous page)

```

--iptables                shaping algorithm. defaults to htb (recommended).
                           use iptables to traffic control.

Routing:
--direction {outgoing,incoming}
                           the direction of network communication that imposes
                           traffic control. 'incoming' requires ifb kernel module
                           and Linux kernel 2.6.20 or later. (default = outgoing)
--network DST_NETWORK, --dst-network DST_NETWORK
                           specify destination IP-address/network that applies
                           traffic control. defaults to any.
--src-network SRC_NETWORK
                           specify source IP-address/network that applies traffic
                           control. defaults to any. this option has no effect
                           when executing with "--direction incoming" option.
                           note: this option required to execute with the
                           --iptables option when using tbf algorithm.
--port DST_PORT, --dst-port DST_PORT
                           specify destination port number that applies traffic
                           control. defaults to any.
--src-port SRC_PORT
                           specify source port number that applies traffic
                           control. defaults to any.
--ipv6
                           apply traffic control to IPv6 packets rather than
                           IPv4.
--exclude-dst-network EXCLUDE_DST_NETWORK
                           exclude a specific destination IP-address/network from
                           a shaping rule.
--exclude-src-network EXCLUDE_SRC_NETWORK
                           exclude a specific source IP-address/network from a
                           shaping rule.
--exclude-dst-port EXCLUDE_DST_PORT
                           exclude a specific destination port from a shaping
                           rule.
--exclude-src-port EXCLUDE_SRC_PORT
                           exclude a specific source port from a shaping rule.

Docker:
--docker                   apply traffic control to a docker container.
--src-container SRC_CONTAINER
--dst-container DST_CONTAINER

Documentation: https://tcconfig.rtfid.io/
Issue tracker: https://github.com/thombashi/tcconfig/issues

```

4.1.2 Basic usage

Examples of outgoing packet traffic control settings are as follows.

4.1.2.1 e.g. Set a limit on bandwidth up to 100Kbps

```
# tcset eth0 --rate 100Kbps
```

4.1.2.2 e.g. Set network latency

You can use time units (such as us/sec/min/etc.) to designate delay time.

Set 100 milliseconds network latency

```
# tcset eth0 --delay 100ms
```

Set 10 seconds network latency

```
# tcset eth0 --delay 10sec
```

Set 0.5 minutes (30 seconds) network latency

```
# tcset eth0 --delay 0.5min
```

You can also use the following time units:

- m/min/mins/minute/minutes
- s/sec/secs/second/seconds
- ms/msec/msecs/millisecond/milliseconds
- us/usec/usecs/microsecond/microseconds

4.1.2.3 e.g. Set 0.1% packet loss

```
# tcset eth0 --loss 0.1%
```

4.1.2.4 e.g. All of the above settings at once

```
# tcset eth0 --rate 100Kbps --delay 100ms --loss 0.1%
```

4.1.2.5 e.g. Specify the IP address of traffic control

```
# tcset eth0 --delay 100ms --network 192.168.0.10
```

4.1.2.6 e.g. Specify the IP network and port of traffic control

```
# tcset eth0 --delay 100ms --network 192.168.0.0/24 --port 80
```

4.1.3 Advanced usage

4.1.3.1 Traffic control of incoming packets

You can set traffic shaping rules to incoming packets by executing `tcset` command with `--direction incoming` option. Other options are the same as in the case of the basic usage.

e.g. Set traffic control for both incoming and outgoing network

```
# tcset eth0 --direction outgoing --rate 200Kbps --network 192.168.0.0/24
# tcset eth0 --direction incoming --rate 1Mbps --network 192.168.0.0/24
```

Requirements

To set incoming packet traffic control requires an additional kernel module named `ifb`, which need to the following conditions:

- Equal or later than Linux kernel version **2.6.20**
- Equal or later than `iproute2` package version **20070313**

4.1.3.2 Set latency distribution

Network latency setting by `--delay` option is a uniform distribution. If you are using `--delay-distro` option, latency decided by a normal distribution.

e.g. Set 100ms +- 20ms network latency with normal distribution

```
# tcset eth0 --delay 100ms --delay-distro 20
```

4.1.3.3 Set multiple traffic shaping rules to an interface

You can set multiple shaping rules to a network interface with `--add` option.

```
tcset eth0 --rate 500Mbps --network 192.168.2.1/32
tcset eth0 --rate 100Mbps --network 192.168.2.2/32 --add
```

4.1.3.4 Using IPv6

IPv6 addresses can be used at `tcset/tcshow` commands with `--ipv6` option.

```
# tcset eth0 --delay 100ms --network 2001:db00::0/24 --ipv6
# tcshow eth0 --ipv6
{
  "eth0": {
    "outgoing": {
      "dst-network=2001:db00::/24, protocol=ipv6": {
        "filter_id": "800::800",
```

(continues on next page)

(continued from previous page)

```

        "delay": "100.0ms",
        "rate": "1Gbps"
    },
    "incoming": {}
}

```

4.1.3.5 Get `tc` commands

You can get `tc` commands to be executed by `tcconfig` commands by executing with `--tc-command` option (Execution of `tcconfig` commands with `--tc-command` option does not affect the traffic rules to the server).

Example

```

$ tcset eth0 --delay 10ms --tc-command
/sbin/tc qdisc add dev eth0 root handle 1: htb default 1
/sbin/tc class add dev eth0 parent 1: classid 1:1 htb rate_
↪1000000kbit
/sbin/tc class add dev eth0 parent 1: classid 1:254 htb rate_
↪1000000Kbit ceil 1000000Kbit
/sbin/tc qdisc add dev eth0 parent 1:254 handle 2873: netem delay 10ms
/sbin/tc filter add dev eth0 protocol ip parent 1: prio 2 u32 match_
↪ip dst 0.0.0.0/0 match ip src 0.0.0.0/0 flowid 1:254

```

4.1.3.6 Generate a `tc` script file

`--tc-script` option generates an executable script which includes `tc` commands to be executed by `tcconfig` commands. The created script can execute at other servers where `tcconfig` not installed (however, you need the `tc` command to run the script).

Example

```

$ tcset eth0 --delay 10ms --tc-script
[INFO] tcconfig: written a tc script to 'tcset_eth0_delay10ms.sh'

(copy the script to a remote server)
$ sudo ./tcset_eth0_delay10ms.sh

```

4.1.3.7 Set a shaping rule for multiple destinations

Example Environment

Multiple hosts (A, B, C, D) are on the same network.

```

A (192.168.0.100) --+---B (192.168.0.2)
                  |
                  +---C (192.168.0.3)
                  |
                  +---D (192.168.0.4)

```


Set a shaping rule to multiple hosts

`--dst-network/--src-network` option can specify not only a host but also network. The following command executed at host A will set a shaping rule that incurs 100 msec network latency to packets from A (192.168.0.100) to specific network (192.168.0.0/28 which include B/C/D).

Example

```
# tcset eth0 --dst-network 192.168.0.0/28 --exclude-dst-network 192.168.
↪0.3 --delay 100ms
```

You can exclude hosts from shaping rules by `--exclude-dst-network/--exclude-src-network` option. The following command executed at host A will set a shaping rule that incurs 100 msec network latency to packets from host A (192.168.0.100) to host B (192.168.0.2)/D (192.168.0.4).

Example

```
# tcset eth0 --dst-network 192.168.0.0/28 --exclude-dst-network 192.168.
↪0.3 --delay 100ms
```

4.1.3.8 Shaping rules for between multiple hosts

Example Environment

Existed multiple networks (192.168.0.0/24, 192.168.1.0/24). Host A (192.168.0.100) and host C (192.168.1.10) belong to a different network. Host B (192.168.0.2/192.168.1.2) belong to both networks.

```
A (192.168.0.100) -- (192.168.0.2) B (192.168.1.2) -- C (192.168.1.10)
```

Set a shaping rule to multiple hosts

The following command executed at host B will set a shaping rule that incurs 100 msec network latency to packets only from host A (192.168.0.100) to host C (192.168.1.10).

Example

```
# tcset eth0 --dst-network 192.168.0.2 --dst-network 192.168.1.2 --delay ↪
↪100ms
```

4.1.4 Docker container

4.1.4.1 Set traffic control to a docker container

Execute `tcconfig` with `--docker` option on a Docker host:

```
# tcset <container name or ID> --docker ...
```

You could use `--src-container/--dst-container` options to specify source/destination container.

4.1.4.2 Set traffic control within a docker container

You need to run a container with `--cap-add NET_ADMIN` option if you would like to set a tc rule within a container:

```
docker run -d --cap-add NET_ADMIN -t <docker image>
```

A container image that builtin tcconfig can be available at <https://hub.docker.com/r/thombashi/tcconfig/>

4.2 Delete traffic control (tcdel command)

tcdel is a command to delete traffic shaping rules from a network interface (device).

Note: tcdel delete mangle tables in iptables. (any other tables are not affected).

4.2.1 tcdel command help

```
usage: tcdel [-h] [-V] [--tc-command | --tc-script] [--debug | --quiet]
           [--stacktrace] [-a] [--id FILTER_ID]
           [--direction {outgoing,incoming}] [--network DST_NETWORK]
           [--src-network SRC_NETWORK] [--port DST_PORT]
           [--src-port SRC_PORT] [--ipv6] [--docker]
           [--src-container SRC_CONTAINER] [--dst-container DST_CONTAINER]
           device

optional arguments:
  -h, --help                show this help message and exit
  -V, --version             show program's version number and exit
  --tc-command              display tc commands to be executed and exit. these
                           commands are not actually executed.
  --tc-script               generate a shell script file that described tc
                           commands. this tc script execution result nearly
                           equivalent with the tcconfig command. the script can
                           be executed without tcconfig package installation.
  --debug                  for debug print.
  --quiet                  suppress execution log messages.

Debug:
  --stacktrace              print stack trace for debug information. --debug
                           option required to see the debug print.

Traffic Control:
  device                   network device name (e.g. eth0)
  -a, --all                delete all of the shaping rules.
  --id FILTER_ID           delete a shaping rule which has a specific id. you can
                           get an id (filter_id) by tcshow command output. e.g.
                           "filter_id": "800::801"

Routing:
  --direction {outgoing,incoming}
                           the direction of network communication that imposes
                           traffic control. 'incoming' requires ifb kernel module
```

(continues on next page)

(continued from previous page)

```

    and Linux kernel 2.6.20 or later. (default = outgoing)
--network DST_NETWORK, --dst-network DST_NETWORK
    specify destination IP-address/network that applies
    traffic control. defaults to any.
--src-network SRC_NETWORK
    specify source IP-address/network that applies traffic
    control. defaults to any. this option has no effect
    when executing with "--direction incoming" option.
    note: this option required to execute with the
    --iptables option when using tbf algorithm.
--port DST_PORT, --dst-port DST_PORT
    specify destination port number that applies traffic
    control. defaults to any.
--src-port SRC_PORT
    specify source port number that applies traffic
    control. defaults to any.
--ipv6
    apply traffic control to IPv6 packets rather than
    IPv4.

Docker:
--docker
    apply traffic control to a docker container.
--src-container SRC_CONTAINER
--dst-container DST_CONTAINER

Documentation: https://tcconfig.rtfid.io/
Issue tracker: https://github.com/thombashi/tcconfig/issues

```

4.2.1.1 e.g. Delete traffic control of eth0

You can delete all of the shaping rules for the eth0 with `-a/--all` option:

```
# tcdel eth0 --all
```

4.2.2 Advanced usage

You can delete a specific shaping rule by either network specifier or `filter_id`.

```

# tcset eth0 --delay 10ms --rate 10k --network 192.168.1.2 --overwrite
# tcset eth0 --delay 100ms --rate 50k --network 192.168.1.3 --add
# tcset eth0 --delay 200ms --rate 100k --network 192.168.0.0/24 --add
# tcshow eth0
{
  "eth0": {
    "outgoing": {
      "dst-network=192.168.1.2/32, protocol=ip": {
        "filter_id": "800::800",
        "delay": "10.0ms",
        "rate": "10Kbps"
      },
      "dst-network=192.168.1.3/32, protocol=ip": {
        "filter_id": "800::801",
        "delay": "100.0ms",
        "rate": "50Kbps"
      }
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

        "dst-network=192.168.0.0/24, protocol=ip": {
            "filter_id": "800::802",
            "delay": "200.0ms",
            "rate": "100Kbps"
        }
    },
    "incoming": {}
}

```

e.g. Delete a shaping rule with network specifier:

```

# tcdel eth0 --dst-network 192.168.1.2
# tcshow eth0
{
  "eth0": {
    "outgoing": {
      "dst-network=192.168.1.3/32, protocol=ip": {
        "filter_id": "800::801",
        "delay": "100.0ms",
        "rate": "50Kbps"
      },
      "dst-network=192.168.0.0/24, protocol=ip": {
        "filter_id": "800::802",
        "delay": "200.0ms",
        "rate": "100Kbps"
      }
    },
    "incoming": {}
  }
}

```

e.g. Delete a shaping rule with filter id:

```

# tcdel eth0 --id 800::801
# tcshow eth0
{
  "eth0": {
    "outgoing": {
      "dst-network=192.168.0.0/24, protocol=ip": {
        "filter_id": "800::802",
        "delay": "200.0ms",
        "rate": "100Kbps"
      }
    },
    "incoming": {}
  }
}

```

4.3 Display traffic control configurations (tcshow command)

tcshow is a command to display the current traffic control settings for network interface(s).

4.3.1 tcshow command help

```
usage: tcshow [-h] [-V] [--tc-command | --tc-script] [--debug | --quiet]
             [--stacktrace] [--ipv6] [--docker] [--color]
             device [device ...]

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  --tc-command          display tc commands to be executed and exit. these commands
                        are not actually executed.
  --tc-script           generate a shell script file that described tc commands. this
                        tc script execution result nearly equivalent with the
                        tcconfig command. the script can be executed without tcconfig
                        package installation.
  --debug              for debug print.
  --quiet              suppress execution log messages.
  --color              colorize the output. require Pygments package.

Debug:
  --stacktrace         print stack trace for debug information. --debug option
                        required to see the debug print.

Traffic Control:
  device              network device name (e.g. eth0)
  --ipv6              Display IPv6 shaping rules. Defaults to show IPv4 shaping
                        rules.

Docker:
  --docker            apply traffic control to a docker container.

Documentation: https://tcconfig.rtfid.io/
Issue tracker: https://github.com/thombashi/tcconfig/issues
```

4.3.1.1 Example

```
# tcset eth0 --delay 10ms --delay-distro 2 --loss 0.01% --rate 0.25Mbps --network_
↪192.168.0.10 --port 8080
# tcset eth0 --delay 1ms --loss 0.02% --rate 500Kbps --direction incoming
# tcshow eth0
{
  "eth0": {
    "outgoing": {
      "dst-network=192.168.0.10/32, dst-port=8080, protocol=ip": {
        "filter_id": "800::800",
        "delay": "10.0ms",
        "delay-distro": "2.0ms",
        "loss": "0.01%",
        "rate": "250Kbps"
      }
    },
    "incoming": {
      "protocol=ip": {
        "filter_id": "800::800",
        "delay": "1.0ms",
        "loss": "0.02%",

```

(continues on next page)

(continued from previous page)

```

        "rate": "500Kbps"
    }
}
}
}

```

Note: Scope of `tcshow` command is limited to parameters that can be set with `tcset` (i.e. `tcshow` is not a general purpose tool to display all of the parameters of the `tc` command).

Note: `tcshow` may output improper values when using `tbfbf`.

4.4 Backup and restore traffic control configurations

`tcshow` command output can be used as a backup, and `tcset` command can restore configurations from a backup.

4.4.1 e.g. Backup configurations

```

# tcset eth0 --delay 10ms --delay-distro 2 --loss 0.01% --rate 0.25Mbps --network_
↳192.168.0.10 --port 8080
# tcset eth0 --delay 1ms --loss 0.02% --rate 500Kbps --direction incoming
# tcset eth1 --delay 2.5ms --delay-distro 1.2 --loss 0.01% --rate 0.25Mbps --port 80
# tcset eth1 --corrupt 0.02% --rate 1.5Mbps --direction incoming --network 192.168.10.
↳0/24

```

Redirect configurations to the `tcconfig.json` file.

```
# tcshow eth0 eth1 > tcconfig.json
```

4.4.2 e.g. Restore configurations

Before restore

```

# tcshow eth0 eth1
{
  "eth1": {
    "outgoing": {},
    "incoming": {}
  },
  "eth0": {
    "outgoing": {},
    "incoming": {}
  }
}

```

Restore from a configuration file (`tcconfig.json`).

```
# tcset tcconfig.json --import-setting
```

After restore

```
# tcshow eth0 eth1
{
  "eth1": {
    "outgoing": {
      "dst-port=80, protocol=ip": {
        "filter_id": "800::800",
        "delay": "2.5ms",
        "delay-distro": "1.2ms",
        "loss": "0.01%",
        "rate": "250Kbps"
      }
    },
    "incoming": {
      "dst-network=192.168.10.0/24, protocol=ip": {
        "filter_id": "800::800",
        "corrupt": "0.02%",
        "rate": "1500Kbps"
      }
    }
  },
  "eth0": {
    "outgoing": {
      "dst-network=192.168.0.10/32, dst-port=8080, protocol=ip": {
        "filter_id": "800::800",
        "delay": "10.0ms",
        "delay-distro": "2.0ms",
        "loss": "0.01%",
        "rate": "250Kbps"
      }
    },
    "incoming": {
      "protocol=ip": {
        "filter_id": "800::800",
        "delay": "1.0ms",
        "loss": "0.02%",
        "rate": "500Kbps"
      }
    }
  }
}
```

4.5 Execute with not super-user

You can execute tcconfig commands with not super-user by using Linux capabilities. Setup Linux capabilities as follows:

```
# the following execution binary paths may different for each environment:
TC_BIN_PATH=/sbin/tc
IP_BIN_PATH=/bin/ip
IPTABLES_BIN_PATH=/sbin/iptables-multi
```

(continues on next page)

(continued from previous page)

```
sudo setcap cap_net_admin+ep $TC_BIN_PATH # mandatory
sudo setcap cap_net_raw,cap_net_admin+ep $IP_BIN_PATH # optional: required to use --
↳direction incoming option
sudo setcap cap_net_raw,cap_net_admin+ep $IPTABLES_BIN_PATH # optional: required to
↳use --iptables option
```

See also:

[capabilities\(7\) - Linux manual page](#)

5.1 RTNETLINK answers: No such file or directory

5.1.1 Phenomenon

tcset command failed with an error message RTNETLINK answers: No such file or directory.

Example

```
# tcset eth0 --rate 1M
[ERROR] tcconfig: RTNETLINK answers: No such file or directory
```

5.1.2 Solution

Load sch_netem module solves the problem.

The cause of the error is sch_netem kernel module not loaded in your system. Execute the following command to load sch_netem module:

```
# modprobe sch_netem
```

If the command failed with the below message, you need to install additional kernel module.

Example - Fedora

```
# modprobe sch_netem
modprobe: FATAL: Module sch_netem not found in directory /lib/modules/4.
↳11.3-202.fc25.x86_64
```

In that case, install kernel-modules-extra package. This package includes the sch_netem module.

Example - Fedora

```
# dnf install kernel-modules-extra
```

Load `sch_netem` module after the package installation.

```
# modprobe sch_netem  
#
```

5.2 RTNETLINK answers: Operation not supported

5.2.1 Phenomenon

`tcset` command with `--direction incoming` failed with an error message `RTNETLINK answers: Operation not supported`.

5.2.2 Solutions

Checking Linux kernel configurations and reconfigure it if required configurations are disabled.

The cause may be some mandatory kernel configurations are disabled. Following configurations are needed to be enabled to use `--direction incoming` option.

- `CONFIG_IP_ADVANCED_ROUTER`
- `CONFIG_IP_MULTIPLE_TABLES`
- `CONFIG_NETFILTER_NETLINK`
- `CONFIG_NETFILTER_NETLINK_QUEUE`
- `CONFIG_NETFILTER_NETLINK_LOG`
- `CONFIG_NF_CT_NETLINK`
- `CONFIG_NETFILTER_XT_TARGET_MARK`
- `CONFIG_NET_SCHED`
- `CONFIG_NET_SCH_INGRESS`
- `CONFIG_SCSI_NETLINK`

e.g. Kernel configurations that enabled the above configurations (Debian)

```
$ cat /boot/config-3.16.0-4-amd64 | egrep "NETFILTER_NETLINK=|NETFILTER_NETLINK_
↪QUEUE=|NETFILTER_NETLINK_LOG=|NF_CT_NETLINK=|SCSI_NETLINK=|IP_ADVANCED_ROUTER=|NET_
↪SCH_INGRESS=|NET_SCHED=|IP_MULTIPLE_TABLES=|NETFILTER_XT_TARGET_MARK="
CONFIG_IP_ADVANCED_ROUTER=y
CONFIG_IP_MULTIPLE_TABLES=y
CONFIG_NETFILTER_NETLINK=m
CONFIG_NETFILTER_NETLINK_QUEUE=m
CONFIG_NETFILTER_NETLINK_LOG=m
CONFIG_NF_CT_NETLINK=m
CONFIG_NETFILTER_XT_TARGET_MARK=m
CONFIG_NET_SCHED=y
CONFIG_NET_SCH_INGRESS=m
CONFIG_SCSI_NETLINK=y
```

These configurations need to either `y` or `m`. If some of the configurations are disabled, you need to:

1. enable the kernel configurations
2. build kernel
3. using the compiled kernel image as boot kernel

Note: Name of the kernel configuration file (`/boot/config-3.16.0-4-amd64`) different depends on the environment.

CHAPTER 6

Indices and tables

- `genindex`

CHAPTER 7

Links

- [pip](#): A tool for installing python packages
- [GitHub repository](#)
- [Issue tracker](#)
- [PyPI](#)

CHAPTER 8

Indices and tables

- genindex