

---

# **tcconfig Documentation**

*Release 0.17.0*

**Tsuyoshi Hombashi**

**Nov 04, 2017**



---

# Table of Contents

---

<b>1</b>	<b>tcconfig</b>	<b>1</b>
1.1	Summary	1
1.2	Traffic control features	1
1.2.1	Traffic shaping target	1
1.2.2	Available parameters	1
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	Installing from PyPI	3
2.2	Installing from files	3
<b>3</b>	<b>Dependencies</b>	<b>5</b>
3.1	Linux packages	5
3.2	Linux kernel module	5
3.3	Python packages	5
3.3.1	Optional	6
3.3.2	Test dependencies	6
<b>4</b>	<b>Usage</b>	<b>7</b>
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	9
4.1.2.3	e.g. Set 0.1% packet loss	10
4.1.2.4	e.g. All of the above 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	10
4.1.3.1	Traffic control of incoming packets	10
4.1.3.2	Set latency distribution	11
4.1.3.3	Set multiple traffic shaping rules per interface	11
4.1.3.4	Using IPv6	11
4.1.3.5	Get <code>tc</code> commands	11
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	12
4.2	Delete traffic control ( <code>tcdel</code> command)	13

4.2.1	tcdel command help . . . . .	13
4.2.1.1	e.g. Delete traffic control of eth0 . . . . .	14
4.2.2	Advanced usage . . . . .	14
4.3	Display traffic control configurations (tcshow command) . . . . .	15
4.3.1	tcshow command help . . . . .	15
4.3.1.1	Example . . . . .	16
4.4	Backup and restore traffic control configurations . . . . .	17
4.4.1	e.g. Backup configurations . . . . .	17
4.4.2	e.g. Restore configurations . . . . .	17
<b>5</b>	<b>Troubleshooting</b>	<b>19</b>
5.1	RTNETLINK answers: No such file or directory . . . . .	19
5.1.1	Phenomenon . . . . .	19
5.1.2	Solution . . . . .	19
5.2	RTNETLINK answers: Operation not supported . . . . .	20
5.2.1	Phenomenon . . . . .	20
5.2.2	Solutions . . . . .	20
<b>6</b>	<b>Indices and tables</b>	<b>23</b>
<b>7</b>	<b>Links</b>	<b>25</b>
<b>8</b>	<b>Indices and tables</b>	<b>27</b>

## 1.1 Summary

A Simple tc command wrapper tool. Easy to set up traffic control of network bandwidth/latency/packet loss/packet-corruption to a network interface.

## 1.2 Traffic control features

### 1.2.1 Traffic shaping target

Apply traffic shaping rules to specific targets:

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

Moreover, exclude from shaping rules from specific targets:

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

### 1.2.2 Available parameters

The following parameters can 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 [%]

### 2.1 Installing from PyPI

`tcconfig` can be installed from [PyPI](#) via `pip` (Python package manager) command.

```
sudo pip install tcconfig
```

### 2.2 Installing from files

The following package includes `tcconfig` and dependency packages. This package is for environments which cannot access to [PyPI](#) directly.

How to install:

1. Navigate to <https://github.com/thombashi/tcconfig/releases/>
2. Download the latest version of `tcconfig_wheel.tar.gz`
3. Copy `tcconfig_wheel.tar.gz` to installation target
4. `tar xvf tcconfig_wheel.tar.gz`
5. `cd tcconfig_wheel/`
6. `./install.sh`





Python 2.7+ or 3.4+

### 3.1 Linux packages

- `iproute/iproute2/iproute-tc` (mandatory: required for `tc` command)
- `iptables` (optional: required to when you use `--iptables` option)

### 3.2 Linux kernel module

- `sch_netem`

### 3.3 Python packages

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

- `DataPropery`
- `ipaddress`
- `logbook`
- `pyparsing`
- `six`
- `subprocrunner`
- `typepy`
- `voluptuous`

### 3.3.1 Optional

- `netifaces`
  - Suppress excessive error messages if this package installed

### 3.3.2 Test dependencies

- `allpairsy`
- `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] [--version] [--tc-command | --tc-script] [--debug | --quiet]
            [--stacktrace] (-d DEVICE | -f CONFIG_FILE)
            [--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,htb}] [--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]
```

optional arguments:

-h, --help	show this help message <b>and</b> exit
--version	show program's version number and exit
--tc-command	display tc commands to be executed <b>and</b> exit. these commands are <b>not</b> executed.
--tc-script	generate a script file that described tc commands which equivalent <b>with</b> execution tcconfig command. the script can be execute without tcconfig package installation.

```

--debug                for debug print.
--quiet                suppress execution log messages.
-d DEVICE, --device DEVICE
                       network device name (e.g. eth0)
-f CONFIG_FILE, --config-file CONFIG_FILE
                       setting traffic controls from a configuration file.
                       output file of the tcshow.
--overwrite            overwrite existing traffic shaping rules.
--change               change existing traffic shaping rules to the new one.
                       this option reduces the shaping rule switching side
                       effect (such as traffic spike) compared to --overwrite
                       option. note: the tcset command 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/milliseconds/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/milliseconds/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,htb}
                       shaping algorithm. defaults to htb (recommended).
--iptables             use iptables to traffic control.

```

## Routing:

```

--direction {outgoing,incoming}
    the direction of network communication that impose
    traffic control. 'incoming' requires Linux kernel
    version 2.6.20 or later. (default = outgoing)
--network DST_NETWORK, --dst-network DST_NETWORK
    target IP-address/network to control traffic
--src-network SRC_NETWORK
    set a traffic shaping rule to specific packets that
    routed from --src-network to --dst-network. this
    option required to execute with the --iptables option
    when you use tbf. the shaping rule only affect to
    outgoing packets (no effect to if you execute with "--
    direction incoming" option)
--port DST_PORT, --dst-port DST_PORT
    target destination port number to control traffic.
--src-port SRC_PORT
    target source port number to control traffic.
--ipv6
    apply traffic control to IPv6 packets rather than
    IPv4.
--exclude-dst-network EXCLUDE_DST_NETWORK
    exclude a shaping rule for a specific destination IP-
    address/network.
--exclude-src-network EXCLUDE_SRC_NETWORK
    exclude a shaping rule for a specific source IP-
    address/network.
--exclude-dst-port EXCLUDE_DST_PORT
    exclude a shaping rule for a specific destination
    port.
--exclude-src-port EXCLUDE_SRC_PORT
    exclude a shaping rule for a specific source port.

```

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 --device eth0 --rate 100k
```

### 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 --device eth0 --delay 100ms
```

### Set 10 seconds network latency

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

### Set 0.5 minutes (30 seconds) network latency

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

#### 4.1.2.3 e.g. Set 0.1% packet loss

```
# tcset --device eth0 --loss 0.1
```

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

```
# tcset --device eth0 --rate 100k --delay 100 --loss 0.1
```

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

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

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

```
# tcset --device eth0 --delay 100 --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 rule 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 --device eth0 --direction outgoing --rate 200K --network 192.168.0.0/24
# tcset --device eth0 --direction incoming --rate 1M --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 --device eth0 --delay 100 --delay-distro 20
```

### 4.1.3.3 Set multiple traffic shaping rules per interface

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

```
tcset --device eth0 --rate 500M --network 192.168.2.0/24
tcset --device eth0 --rate 100M --network 192.168.0.0/24 --add
```

### 4.1.3.4 Using IPv6

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

```
# tcset --device eth0 --delay 100 --network 2001:db00::0/24 --ipv6
# tcshow --device eth0 --ipv6
{
  "eth0": {
    "outgoing": {
      "dst-network=2001:db00::/24, protocol=ipv6": {
        "delay": "100.0",
        "rate": "1G"
      }
    },
    "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 (no `tc` configuration have made to the execution server by this command).

#### Example

```
# tcset --device eth0 --delay 10 --tc-command
tc qdisc add dev eth0 root handle 1f87: htb default 1
tc class add dev eth0 parent 1f87: classid 1f87:1 htb rate 1000000kbit
tc class add dev eth0 parent 1f87: classid 1f87:2 htb rate 1000000Kbit
↪ceil 1000000Kbit
tc qdisc add dev eth0 parent 1f87:2 handle 2007: netem delay 10.0ms
tc filter add dev eth0 protocol ip parent 1f87: prio 1 u32 match ip dst
↪0.0.0.0/0 flowid 1f87:2
```

#### 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 --device eth0 --delay 10 --tc-script
[INFO] tcconfig: written a tc script to 'tcset_eth0.sh'
# ./tcset_eth0.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 -d eth0 --dst-network 192.168.0.0/28 --exclude-dst-network 192.
↳168.0.3 --delay 100
```

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 -d eth0 --dst-network 192.168.0.0/28 --exclude-dst-network 192.
↳168.0.3 --delay 100
```

#### 4.1.3.8 Shaping rules for between multiple hosts

##### Example Environment

Existed multiple networks (192.168.0.0/24, 192.168.10.1/24). Host A (192.168.0.100) and host C (192.168.0.100) 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 -d eth0 --dst-network 192.168.0.2 --dst-network 192.168.1.2 --
↳delay 100
```

## 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] [--version] [--tc-command | --tc-script] [--debug | --quiet]
            [--stacktrace] -d DEVICE [-a] [--id FILTER_ID]
            [--direction {outgoing,incoming}] [--network DST_NETWORK]
            [--src-network SRC_NETWORK] [--port DST_PORT]
            [--src-port SRC_PORT] [--ipv6]

optional arguments:
  -h, --help                show this help message and exit
  --version                 show program's version number and exit
  --tc-command              display tc commands to be executed and exit. these
                           commands are not executed.
  --tc-script               generate a script file that described tc commands
                           which equivalent with execution tcconfig command. the
                           script can be execute 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:
  -d DEVICE, --device 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 impose
    traffic control. 'incoming' requires Linux kernel
    version 2.6.20 or later. (default = outgoing)
--network DST_NETWORK, --dst-network DST_NETWORK
    target IP-address/network to control traffic
--src-network SRC_NETWORK
    set a traffic shaping rule to specific packets that
    routed from --src-network to --dst-network. this
    option required to execute with the --iptables option
    when you use tbf. the shaping rule only affect to
    outgoing packets (no effect to if you execute with "--
    direction incoming" option)
--port DST_PORT, --dst-port DST_PORT
    target destination port number to control traffic.
--src-port SRC_PORT
    target source port number to control traffic.
--ipv6
    apply traffic control to IPv6 packets rather than
    IPv4.

```

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 --device eth0 --all
```

## 4.2.2 Advanced usage

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

```

# tcset --device ens33 --delay 10 --rate 10k --network 192.168.1.2 --overwrite
# tcset --device ens33 --delay 100 --rate 50k --network 192.168.1.3 --add
# tcset --device ens33 --delay 200 --rate 100k --network 192.168.0.0/24 --add
# tcshow -d ens33
{
  "ens33": {
    "outgoing": {
      "dst-network=192.168.1.2/32, protocol=ip": {
        "delay": 10,
        "rate": "10K",
        "filter_id": "800::800"
      },
      "dst-network=192.168.0.0/24, protocol=ip": {
        "delay": 200,
        "rate": "100K",
        "filter_id": "800::802"
      },
      "dst-network=192.168.1.3/32, protocol=ip": {
        "delay": 100,
        "rate": "50K",
        "filter_id": "800::801"
      }
    }
  }
},

```

```

    "incoming": {}
  }
}

```

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

```

# tcdel --device ens33 --dst-network 192.168.1.2
# tcshow -d ens33
{
  "ens33": {
    "outgoing": {
      "dst-network=192.168.0.0/24, protocol=ip": {
        "delay": 200,
        "rate": "100K",
        "filter_id": "800::802"
      },
      "dst-network=192.168.1.3/32, protocol=ip": {
        "delay": 100,
        "rate": "50K",
        "filter_id": "800::801"
      }
    },
    "incoming": {}
  }
}

```

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

```

# tcdel --device ens33 --id 800::801
# tcshow -d ens33
{
  "ens33": {
    "outgoing": {
      "dst-network=192.168.0.0/24, protocol=ip": {
        "delay": 200,
        "rate": "100K",
        "filter_id": "800::802"
      }
    },
    "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] [--version] [--tc-command | --tc-script]
             [--debug | --quiet] [--stacktrace] -d DEVICE [--ipv6]

```

optional arguments:

```

  -h, --help                show this help message and exit

```

```
--version          show program's version number and exit
--tc-command       display tc commands to be executed and exit. these
                  commands are not executed.
--tc-script        generate a script file that described tc commands
                  which equivalent with execution tcconfig command. the
                  script can be execute 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:

```
-d DEVICE, --device DEVICE      network device name (e.g. eth0)
--ipv6                          Display IPv6 shaping rules. Defaults to show IPv4
                                shaping rules.
```

Issue tracker: <https://github.com/thombashi/tcconfig/issues>

### 4.3.1.1 Example

```
# tcset --device eth0 --delay 10 --delay-distro 2 --loss 0.01 --rate 0.25M --network_
↪192.168.0.10 --port 8080
# tcset --device eth0 --delay 1 --loss 0.02 --rate 500K --direction incoming
# tcshow --device eth0
{
  "eth0": {
    "outgoing": {
      "dst-network=192.168.0.10/32, dst-port=8080": {
        "delay": "10.0",
        "loss": "0.01",
        "rate": "250K",
        "delay-distro": "2.0"
      },
      "dst-network=0.0.0.0/0": {}
    },
    "incoming": {
      "dst-network=0.0.0.0/0": {
        "delay": "1.0",
        "loss": "0.02",
        "rate": "500K"
      }
    }
  }
}
```

---

**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 --device eth0 --delay 10 --delay-distro 2 --loss 0.01 --rate 0.25M --network_
↪192.168.0.10 --port 8080
# tcset --device eth0 --delay 1 --loss 0.02 --rate 500K --direction incoming
# tcset --device eth1 --delay 2.5 --delay-distro 1.2 --loss 0.01 --rate 0.25M --port_
↪80
# tcset --device eth1 --corrupt 0.02 --rate 1.5M --direction incoming --network 192.
↪168.10.0/24
```

Redirect configurations to the `tcconfig.json` file.

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

### 4.4.2 e.g. Restore configurations

Before restore

```
# tcshow --device eth0 --device eth1
{
  "eth1": {
    "outgoing": {},
    "incoming": {}
  },
  "eth0": {
    "outgoing": {},
    "incoming": {}
  }
}
```

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

```
# tcset -f tcconfig.json
```

After restore

```
# tcshow --device eth0 --device eth1
{
  "eth1": {
    "outgoing": {
      "dst-port=80": {
        "delay": "2.5",
        "loss": "0.01",
        "rate": "250K",
        "delay-distro": "1.2"
      }
    }
  }
}
```

```
    },
    "dst-network=0.0.0.0/0": {}
  },
  "incoming": {
    "dst-network=192.168.10.0/24": {
      "corrupt": "0.02",
      "rate": "1500K"
    },
    "dst-network=0.0.0.0/0": {}
  }
},
"eth0": {
  "outgoing": {
    "dst-network=192.168.0.10/32, dst-port=8080": {
      "delay": "10.0",
      "loss": "0.01",
      "rate": "250K",
      "delay-distro": "2.0"
    },
    "dst-network=0.0.0.0/0": {}
  },
  "incoming": {
    "dst-network=0.0.0.0/0": {
      "delay": "1.0",
      "loss": "0.02",
      "rate": "500K"
    }
  }
}
}
```

## 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 --device 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