Documentation for /proc/sys/net/

This file contains the documentation for the sysctl files in /proc/sys/net

The interface to the networking parts of the kernel is located in /proc/sys/net. The following table shows all possible subdirectories. You may see only some of them, depending on your kernel's configuration.

Table : Subdirectories in /proc/sys/net

Directory	Content	Directory	Content
core	General parameter	appletalk	Appletalk protocol
unix	Unix domain sockets	netrom	NET/ROM
802	E802 protocol	ax25	AX25
ethernet	Ethernet protocol	rose	X.25 PLP layer
ipv4	IP version 4	x25	X.25 protocol
bridge	Bridging	decnet	DEC net
ipv6	IP version 6	tipc	TIPC

1. /proc/sys/net/core - Network core options

bpf_jit_enable

This enables the BPF Just in Time (JIT) compiler. BPF is a flexible and efficient infrastructure allowing to execute bytecode at various hook points. It is used in a number of Linux kernel subsystems such as networking (e.g. XDP, tc), tracing (e.g. kprobes, uprobes, tracepoints) and security (e.g. seccomp). LLVM has a BPF back end that can compile restricted C into a sequence of BPF instructions. After program load through bpf(2) and passing a verifier in the kernel, a JIT will then translate these BPF proglets into native CPU instructions. There are two flavors of JITs, the newer eBPF JIT currently supported on:

- x86_64
- x86_32
- arm64
- arm32
- ppc64
- ppc32
- sparc64
- mips64
- s390x
- riscv64
- riscv32

And the older cBPF JIT supported on the following archs:

- mips
- sparc

eBPF JITs are a superset of cBPF JITs, meaning the kernel will migrate cBPF instructions into eBPF instructions and then JIT compile them transparently. Older cBPF JITs can only translate tcpdump filters, seccomp rules, etc, but not mentioned eBPF programs loaded through bpf(2).

Values:

- 0 - disable the JIT (default value)
- 1 - enable the JIT
- 2 - enable the JIT and ask the compiler to emit traces on kernel log.

bpf_jit_harden

This enables hardening for the BPF JIT compiler. Supported are eBPF JIT backends. Enabling hardening trades off performance, but can mitigate JIT spraying.

Values:

- 0 - disable JIT hardening (default value)
- 1 - enable JIT hardening for unprivileged users only
- 2 - enable JIT hardening for all users

bpf_jit_kallsyms

When BPF JIT compiler is enabled, then compiled images are unknown addresses to the kernel, meaning they neither show up in traces nor in /proc/kallsyms. This enables export of these addresses, which can be used for debugging/tracing. If bpf_jit_harden is enabled, this feature is disabled.

Values :

- 0 - disable JIT kallsyms export (default value)
- 1 - enable JIT kallsyms export for privileged users only

bpf_jit_limit

This enforces a global limit for memory allocations to the BPF JIT compiler in order to reject unprivileged JIT requests once it has been surpassed. bpf_jit_limit contains the value of the global limit in bytes.

dev_weight

The maximum number of packets that kernel can handle on a NAPI interrupt, it's a Per-CPU variable. For drivers that support LRO or GRO_HW, a hardware aggregated packet is counted as one packet in this context.

Default: 64

dev_weight_rx_bias

RPS (e.g. RFS, aRFS) processing is competing with the registered NAPI poll function of the driver for the per softirq cycle netdev_budget. This parameter influences the proportion of the configured netdev_budget that is spent on RPS based packet processing during RX softirq cycles. It is further meant for making current dev_weight adaptable for asymmetric CPU needs on RX/TX side of the network stack. (see dev_weight_tx_bias) It is effective on a per CPU basis. Determination is based on dev_weight and is calculated multiplicative (dev_weight * dev_weight_rx_bias).

Default: 1

dev_weight_tx_bias

Scales the maximum number of packets that can be processed during a TX softirq cycle. Effective on a per CPU basis. Allows scaling of current dev_weight for asymmetric net stack processing needs. Be careful to avoid making TX softirq processing a CPU hog.

Calculation is based on dev_weight (dev_weight * dev_weight_tx_bias).

Default: 1

default_qdisc

The default queuing discipline to use for network devices. This allows overriding the default of pfifo_fast with an alternative. Since the default queuing discipline is created without additional parameters so is best suited to queuing disciplines that work well without configuration like stochastic fair queue (sfq), CoDel (codel) or fair queue CoDel (fq_codel). Don't use queuing disciplines like Hierarchical Token Bucket or Deficit Round Robin which require setting up classes and bandwidths. Note that physical multiqueue interfaces still use mq as root qdisc, which in turn uses this default for its leaves. Virtual devices (like e.g. lo or veth) ignore this setting and instead default to noqueue.

Default: pfifo_fast

busy_read

Low latency busy poll timeout for socket reads. (needs CONFIG_NET_RX_BUSY_POLL) Approximate time in us to busy loop waiting for packets on the device queue. This sets the default value of the SO_BUSY_POLL socket option. Can be set or overridden per socket by setting socket option SO_BUSY_POLL, which is the preferred method of enabling. If you need to enable the feature globally via sysctl, a value of 50 is recommended.

Will increase power usage.

Default: 0 (off)

busy_poll

Low latency busy poll timeout for poll and select. (needs CONFIG_NET_RX_BUSY_POLL) Approximate time in us to busy loop waiting for events. Recommended value depends on the number of sockets you poll on. For several sockets 50, for several hundreds 100. For more than that you probably want to use epoll. Note that only sockets with SO_BUSY_POLL set will be busy polled, so you want to either selectively set SO_BUSY_POLL on those sockets or set sysctl.net.busy_read globally.

Will increase power usage.

Default: 0 (off)

rmem_default

The default setting of the socket receive buffer in bytes.

rmem_max

The maximum receive socket buffer size in bytes.

tstamp_allow_data

Allow processes to receive tx timestamps looped together with the original packet contents. If disabled, transmit timestamp requests from unprivileged processes are dropped unless socket option SOF_TIMESTAMPING_OPT_TSONLY is set.

Default: 1 (on)

wmem_default

The default setting (in bytes) of the socket send buffer.

wmem_max

The maximum send socket buffer size in bytes.

message_burst and message_cost

These parameters are used to limit the warning messages written to the kernel log from the networking code. They enforce a rate limit to make a denial-of-service attack impossible. A higher message_cost factor, results in fewer messages that will be written. Message_burst controls when messages will be dropped. The default settings limit warning messages to one every five seconds.

warnings

This sysctl is now unused.

This was used to control console messages from the networking stack that occur because of problems on the network like duplicate address or bad checksums.

These messages are now emitted at KERN_DEBUG and can generally be enabled and controlled by the dynamic_debug facility.

netdev_budget

Maximum number of packets taken from all interfaces in one polling cycle (NAPI poll). In one polling cycle interfaces which are registered to polling are probed in a round-robin manner. Also, a polling cycle may not exceed netdev_budget_usecs microseconds, even if netdev_budget has not been exhausted.

netdev_budget_usecs

Maximum number of microseconds in one NAPI polling cycle. Polling will exit when either netdev_budget_usecs have elapsed during the poll cycle or the number of packets processed reaches netdev_budget.

netdev_max_backlog

Maximum number of packets, queued on the INPUT side, when the interface receives packets faster than kernel can process them.

netdev_rss_key

RSS (Receive Side Scaling) enabled drivers use a 40 bytes host key that is randomly generated. Some user space might need to gather its content even if drivers do not provide ethtool -x support yet.

```
myhost:~# cat /proc/sys/net/core/netdev_rss_key
84:50:f4:00:a8:15:d1:a7:e9:7f:1d:60:35:c7:47:25:42:97:74:ca:56:bb:b6:a1:d8: ... (52 bytes total)
```

File contains nul bytes if no driver ever called netdev_rss_key_fill() function.

Note:

/proc/sys/net/core/netdev_rss_key contains 52 bytes of key, but most drivers only use 40 bytes of it.

```
myhost:~# ethtool -x eth0
RX flow hash indirection table for eth0 with 8 RX ring(s):
    0:    0     1     2     3     4     5     6     7
RSS hash key:
84:50:f4:00:a8:15:d1:a7:e9:7f:1d:60:35:c7:47:25:42:97:74:ca:56:bb:b6:a1:d8:43:e3:c9:0c:fd:17:55:c2:3a:4d:69:ed:
```

netdev_tstamp_prequeue

If set to 0, RX packet timestamps can be sampled after RPS processing, when the target CPU processes packets. It might give some delay on timestamps, but permit to distribute the load on several cpus.

If set to 1 (default), timestamps are sampled as soon as possible, before queueing.

netdev_unregister_timeout_secs

Unregister network device timeout in seconds. This option controls the timeout (in seconds) used to issue a warning while waiting for a network device refcount to drop to 0 during device unregistration. A lower value may be useful during bisection to detect a leaked reference faster. A larger value may be useful to prevent false warnings on slow/loaded systems. Default value is 10, minimum 1, maximum 3600.

optmem_max

Maximum ancillary buffer size allowed per socket. Ancillary data is a sequence of struct cmsghdr structures with appended data.

fb_tunnels_only_for_init_net

Controls if fallback tunnels (like tunl0, gre0, gretap0, erspan0, sit0, ip6tnl0, ip6gre0) are automatically created. There are 3 possibilities (a) value = 0; respective fallback tunnels are created when module is loaded in every net namespaces (backward compatible behavior). (b) value = 1; [kcmd value: initns] respective fallback tunnels are created only in init net namespace and every other net namespace will not have them. (c) value = 2; [kcmd value: none] fallback tunnels are not created when a module is loaded in any of the net namespace. Setting value to "2" is pointless after boot if these modules are built-in, so there is a kernel command-line option that can change this default. Please refer to Documentation/admin-guide/kernel-parameters.txt for additional details.

Not creating fallback tunnels gives control to userspace to create whatever is needed only and avoid creating devices which are redundant.

Default : 0 (for compatibility reasons)

devconf_inherit_init_net

Controls if a new network namespace should inherit all current settings under /proc/sys/net/{ipv4,ipv6}/conf/{all,default}/. By default, we keep the current behavior: for IPv4 we inherit all current settings from init_net and for IPv6 we reset all settings to default.

If set to 1, both IPv4 and IPv6 settings are forced to inherit from current ones in init_net. If set to 2, both IPv4 and IPv6 settings are forced to reset to their default values. If set to 3, both IPv4 and IPv6 settings are forced to inherit from current ones in the netns where this new netns has been created.

Default : 0 (for compatibility reasons)

txrehash

Controls default hash rethink behaviour on listening socket when SO_TXREHASH option is set to SOCK_TXREHASH_DEFAULT (i. e. not overridden by setsockopt).

If set to 1 (default), hash rethink is performed on listening socket. If set to 0, hash rethink is not performed.

2. /proc/sys/net/unix - Parameters for Unix domain sockets

There is only one file in this directory. unix_dgram_qlen limits the max number of datagrams queued in Unix domain socket's buffer. It will not take effect unless PF_UNIX flag is specified.

3. /proc/sys/net/ipv4 - IPV4 settings

Please see: Documentation/networking/ip-sysctl.rst and Documentation/admin-guide/sysctl/net.rst for descriptions of these entries.

4. Appletalk

The /proc/sys/net/appletalk directory holds the Appletalk configuration data when Appletalk is loaded. The configurable parameters are:

aarp-expiry-time

The amount of time we keep an ARP entry before expiring it. Used to age out old hosts.

aarp-resolve-time

The amount of time we will spend trying to resolve an Appletalk address.

aarp-retransmit-limit

The number of times we will retransmit a query before giving up.

aarp-tick-time

Controls the rate at which expires are checked.

The directory /proc/net/appletalk holds the list of active Appletalk sockets on a machine.

The fields indicate the DDP type, the local address (in network:node format) the remote address, the size of the transmit pending queue, the size of the received queue (bytes waiting for applications to read) the state and the uid owning the socket.

/proc/net/atalk_iface lists all the interfaces configured for appletalk.It shows the name of the interface, its Appletalk address, the network range on that address (or network number for phase 1 networks), and the status of the interface.

/proc/net/atalk_route lists each known network route. It lists the target (network) that the route leads to, the router (may be directly connected), the route flags, and the device the route is using.

5. TIPC

tipc_rmem

The TIPC protocol now has a tunable for the receive memory, similar to the tcp_rmem - i.e. a vector of 3 INTEGERs: (min, default, max)

```
# cat /proc/sys/net/tipc/tipc_rmem
4252725 34021800        68043600
#
```

The max value is set to CONN_OVERLOAD_LIMIT, and the default and min values are scaled (shifted) versions of that same value. Note that the min value is not at this point in time used in any meaningful way, but the triplet is preserved in order to be

consistent with things like tcp_rmem.

named_timeout

TIPC name table updates are distributed asynchronously in a cluster, without any form of transaction handling. This means that different race scenarios are possible. One such is that a name withdrawal sent out by one node and received by another node may arrive after a second, overlapping name publication already has been accepted from a third node, although the conflicting updates originally may have been issued in the correct sequential order. If named_timeout is nonzero, failed topology updates will be placed on a defer queue until another event arrives that clears the error, or until the timeout expires. Value is in milliseconds.