```
/proc/sys/net/ipv4/* Variables:

ip_forward - BOOLEAN
        0 - disabled (default)
        not 0 - enabled

        Forward Packets between interfaces.

        This variable is special, its change resets all configuration
        parameters to their default state (RFC1122 for hosts, RFC1812
        for routers)

ip_default_ttl - INTEGER
        default 64

ip_no_pmtu_disc - BOOLEAN
        Disable Path MTU Discovery.
        default FALSE

min_pmtu - INTEGER
        default 562 - minimum discovered Path MTU

mtu_expires - INTEGER
        Time, in seconds, that cached PMTU information is kept.

min_adv_mss - INTEGER
        The advertised MSS depends on the first hop route MTU, but will
        never be lower than this setting.

rt_cache_rebuild_count - INTEGER
        The per net-namespace route cache emergency rebuild threshold.
        Any net-namespace having its route cache rebuilt due to
        a hash bucket chain being too long more than this many times
        will have its route caching disabled

IP Fragmentation:

ipfrag_high_thresh - INTEGER
        Maximum memory used to reassemble IP fragments. When
        ipfrag_high_thresh bytes of memory is allocated for this purpose,
        the fragment handler will toss packets until ipfrag_low_thresh
        is reached.

ipfrag_low_thresh - INTEGER
        See ipfrag_high_thresh

ipfrag_time - INTEGER
        Time in seconds to keep an IP fragment in memory.

ipfrag_secret_interval - INTEGER
        Regeneration interval (in seconds) of the hash secret (or lifetime
        for the hash secret) for IP fragments.
        Default: 600

ipfrag_max_dist - INTEGER
        ipfrag_max_dist is a non-negative integer value which defines the
```

maximum "disorder" which is allowed among fragments which share a common IP source address. Note that reordering of packets is not unusual, but if a large number of fragments arrive from a source IP address while a particular fragment queue remains incomplete, it probably indicates that one or more fragments belonging to that queue have been lost. When ipfrag_max_dist is positive, an additional check is done on fragments before they are added to a reassembly queue - if ipfrag_max_dist (or more) fragments have arrived from a particular IP address between additions to any IP fragment queue using that source address, it's presumed that one or more fragments in the queue are lost. The existing fragment queue will be dropped, and a new one started. An ipfrag_max_dist value of zero disables this check.

Using a very small value, e.g. 1 or 2, for ipfrag_max_dist can result in unnecessarily dropping fragment queues when normal reordering of packets occurs, which could lead to poor application performance. Using a very large value, e.g. 50000, increases the likelihood of incorrectly reassembling IP fragments that originate from different IP datagrams, which could result in data corruption.
Default: 64

INET peer storage:

inet_peer_threshold - INTEGER

The approximate size of the storage. Starting from this threshold entries will be thrown aggressively. This threshold also determines entries' time-to-live and time intervals between garbage collection passes. More entries, less time-to-live, less GC interval.

inet_peer_minttl - INTEGER

Minimum time-to-live of entries. Should be enough to cover fragment time-to-live on the reassembling side. This minimum time-to-live is guaranteed if the pool size is less than inet_peer_threshold.
Measured in seconds.

inet_peer_maxttl - INTEGER

Maximum time-to-live of entries. Unused entries will expire after this period of time if there is no memory pressure on the pool (i.e. when the number of entries in the pool is very small).
Measured in seconds.

inet_peer_gc_mintime - INTEGER

Minimum interval between garbage collection passes. This interval is in effect under high memory pressure on the pool.
Measured in seconds.

inet_peer_gc_maxtime - INTEGER

Minimum interval between garbage collection passes. This interval is in effect under low (or absent) memory pressure on the pool.
Measured in seconds.

TCP variables:

somaxconn - INTEGER

Limit of socket listen() backlog, known in userspace as SOMAXCONN. Defaults to 128. See also tcp_max_syn_backlog for additional tuning

```
	for TCP sockets.

tcp_abc - INTEGER
	Controls Appropriate Byte Count (ABC) defined in RFC3465.
	ABC is a way of increasing congestion window (cwnd) more slowly
	in response to partial acknowledgments.
	Possible values are:
		0 increase cwnd once per acknowledgment (no ABC)
		1 increase cwnd once per acknowledgment of full sized segment
		2 allow increase cwnd by two if acknowledgment is
		  of two segments to compensate for delayed acknowledgments.
	Default: 0 (off)

tcp_abort_on_overflow - BOOLEAN
	If listening service is too slow to accept new connections,
	reset them. Default state is FALSE. It means that if overflow
	occurred due to a burst, connection will recover. Enable this
	option _only_ if you are really sure that listening daemon
	cannot be tuned to accept connections faster. Enabling this
	option can harm clients of your server.

tcp_adv_win_scale - INTEGER
	Count buffering overhead as bytes/2^tcp_adv_win_scale
	(if tcp_adv_win_scale > 0) or bytes-bytes/2^(-tcp_adv_win_scale),
	if it is <= 0.
	Default: 2

tcp_allowed_congestion_control - STRING
	Show/set the congestion control choices available to non-privileged
	processes. The list is a subset of those listed in
	tcp_available_congestion_control.
	Default is "reno" and the default setting (tcp_congestion_control).

tcp_app_win - INTEGER
	Reserve max(window/2^tcp_app_win, mss) of window for application
	buffer. Value 0 is special, it means that nothing is reserved.
	Default: 31

tcp_available_congestion_control - STRING
	Shows the available congestion control choices that are registered.
	More congestion control algorithms may be available as modules,
	but not loaded.

tcp_base_mss - INTEGER
	The initial value of search_low to be used by the packetization layer
	Path MTU discovery (MTU probing).  If MTU probing is enabled,
	this is the initial MSS used by the connection.

tcp_congestion_control - STRING
	Set the congestion control algorithm to be used for new
	connections. The algorithm "reno" is always available, but
	additional choices may be available based on kernel configuration.
	Default is set as part of kernel configuration.

tcp_cookie_size - INTEGER
	Default size of TCP Cookie Transactions (TCPCT) option, that may be
	overridden on a per socket basis by the TCPCT socket option.
	Values greater than the maximum (16) are interpreted as the maximum.
	Values greater than zero and less than the minimum (8) are interpreted
	as the minimum.  Odd values are interpreted as the next even value.
	Default: 0 (off).

tcp_dsack - BOOLEAN
	Allows TCP to send "duplicate" SACKs.

tcp_ecn - BOOLEAN
	Enable Explicit Congestion Notification (ECN) in TCP. ECN is only
	used when both ends of the TCP flow support it. It is useful to
	avoid losses due to congestion (when the bottleneck router supports
	ECN).
	Possible values are:
		0 disable ECN
		1 ECN enabled
		2 Only server-side ECN enabled. If the other end does
		  not support ECN, behavior is like with ECN disabled.
	Default: 2

tcp_fack - BOOLEAN
	Enable FACK congestion avoidance and fast retransmission.
	The value is not used, if tcp_sack is not enabled.

tcp_fin_timeout - INTEGER
	Time to hold socket in state FIN-WAIT-2, if it was closed
	by our side. Peer can be broken and never close its side,
	or even died unexpectedly. Default value is 60sec.
	Usual value used in 2.2 was 180 seconds, you may restore
	it, but remember that if your machine is even underloaded WEB server,
	you risk to overflow memory with kilotons of dead sockets,
	FIN-WAIT-2 sockets are less dangerous than FIN-WAIT-1,
	because they eat maximum 1.5K of memory, but they tend
	to live longer. Cf. tcp_max_orphans.

tcp_frto - INTEGER
	Enables Forward RTO-Recovery (F-RTO) defined in RFC4138.
	F-RTO is an enhanced recovery algorithm for TCP retransmission
	timeouts.  It is particularly beneficial in wireless environments
	where packet loss is typically due to random radio interference
	rather than intermediate router congestion.  F-RTO is sender-side
	only modification. Therefore it does not require any support from
	the peer.

	If set to 1, basic version is enabled.  2 enables SACK enhanced
	F-RTO if flow uses SACK.  The basic version can be used also when
	SACK is in use though scenario(s) with it exists where F-RTO
	interacts badly with the packet counting of the SACK enabled TCP
	flow.

tcp_frto_response - INTEGER
	When F-RTO has detected that a TCP retransmission timeout was
	spurious (i.e, the timeout would have been avoided had TCP set a
	longer retransmission timeout), TCP has several options what to do
	next. Possible values are:
```

0 Rate halving based; a smooth and conservative response, results in halved cwnd and ssthresh after one RTT
1 Very conservative response; not recommended because even though being valid, it interacts poorly with the rest of Linux TCP, halves cwnd and ssthresh immediately
2 Aggressive response; undoes congestion control measures that are now known to be unnecessary (ignoring the possibility of a lost retransmission that would require TCP to be more cautious), cwnd and ssthresh are restored to the values prior timeout

Default: 0 (rate halving based)

tcp_keepalive_time - INTEGER

How often TCP sends out keepalive messages when keepalive is enabled.
Default: 2hours.

tcp_keepalive_probes - INTEGER

How many keepalive probes TCP sends out, until it decides that the connection is broken. Default value: 9.

tcp_keepalive_intvl - INTEGER

How frequently the probes are send out. Multiplied by tcp_keepalive_probes it is time to kill not responding connection, after probes started. Default value: 75sec i.e. connection will be aborted after ~11 minutes of retries.

tcp_low_latency - BOOLEAN

If set, the TCP stack makes decisions that prefer lower latency as opposed to higher throughput. By default, this option is not set meaning that higher throughput is preferred. An example of an application where this default should be changed would be a Beowulf compute cluster.
Default: 0

tcp_max_orphans - INTEGER

Maximal number of TCP sockets not attached to any user file handle, held by system. If this number is exceeded orphaned connections are reset immediately and warning is printed. This limit exists only to prevent simple DoS attacks, you _must_ not rely on this or lower the limit artificially, but rather increase it (probably, after increasing installed memory), if network conditions require more than default value, and tune network services to linger and kill such states more aggressively. Let me to remind again: each orphan eats up to ~64K of unswappable memory.

tcp_max_syn_backlog - INTEGER

Maximal number of remembered connection requests, which are still did not receive an acknowledgment from connecting client. Default value is 1024 for systems with more than 128Mb of memory, and 128 for low memory machines. If server suffers of overload, try to increase this number.

tcp_max_tw_buckets - INTEGER

Maximal number of timewait sockets held by system simultaneously. If this number is exceeded time-wait socket is immediately destroyed and warning is printed. This limit exists only to prevent simple DoS attacks, you _must_ not lower the limit artificially, but rather increase it (probably, after increasing installed memory), if network conditions require more than default value.

tcp_mem - vector of 3 INTEGERs: min, pressure, max

min: below this number of pages TCP is not bothered about its memory appetite.

pressure: when amount of memory allocated by TCP exceeds this number of pages, TCP moderates its memory consumption and enters memory pressure mode, which is exited when memory consumption falls under "min".

max: number of pages allowed for queueing by all TCP sockets.

Defaults are calculated at boot time from amount of available memory.

tcp_moderate_rcvbuf - BOOLEAN

If set, TCP performs receive buffer auto-tuning, attempting to automatically size the buffer (no greater than tcp_rmem[2]) to match the size required by the path for full throughput. Enabled by default.

tcp_mtu_probing - INTEGER

Controls TCP Packetization-Layer Path MTU Discovery. Takes three values:

```
  0 - Disabled
  1 - Disabled by default, enabled when an ICMP black hole detected
  2 - Always enabled, use initial MSS of tcp_base_mss.
```

tcp_no_metrics_save - BOOLEAN

By default, TCP saves various connection metrics in the route cache when the connection closes, so that connections established in the near future can use these to set initial conditions. Usually, this increases overall performance, but may sometimes cause performance degradation. If set, TCP will not cache metrics on closing connections.

tcp_orphan_retries - INTEGER

This value influences the timeout of a locally closed TCP connection, when RTO retransmissions remain unacknowledged.
See tcp_retries2 for more details.

The default value is 7.
If your machine is a loaded WEB server,
you should think about lowering this value, such sockets
may consume significant resources. Cf. tcp_max_orphans.

tcp_reordering - INTEGER

Maximal reordering of packets in a TCP stream.
Default: 3

tcp_retrans_collapse - BOOLEAN

Bug-to-bug compatibility with some broken printers.

On retransmit try to send bigger packets to work around bugs in certain TCP stacks.

tcp_retries1 - INTEGER

This value influences the time, after which TCP decides, that something is wrong due to unacknowledged RTO retransmissions, and reports this suspicion to the network layer.
See tcp_retries2 for more details.

RFC 1122 recommends at least 3 retransmissions, which is the default.

tcp_retries2 - INTEGER

This value influences the timeout of an alive TCP connection, when RTO retransmissions remain unacknowledged.
Given a value of N, a hypothetical TCP connection following exponential backoff with an initial RTO of TCP_RTO_MIN would retransmit N times before killing the connection at the (N+1)th RTO.

The default value of 15 yields a hypothetical timeout of 924.6 seconds and is a lower bound for the effective timeout.
TCP will effectively time out at the first RTO which exceeds the hypothetical timeout.

RFC 1122 recommends at least 100 seconds for the timeout, which corresponds to a value of at least 8.

tcp_rfc1337 - BOOLEAN

If set, the TCP stack behaves conforming to RFC1337. If unset, we are not conforming to RFC, but prevent TCP TIME_WAIT assassination.
Default: 0

tcp_rmem - vector of 3 INTEGERs: min, default, max

min: Minimal size of receive buffer used by TCP sockets.
It is guaranteed to each TCP socket, even under moderate memory pressure.
Default: 8K

default: initial size of receive buffer used by TCP sockets.
This value overrides net.core.rmem_default used by other protocols.
Default: 87380 bytes. This value results in window of 65535 with default setting of tcp_adv_win_scale and tcp_app_win:0 and a bit less for default tcp_app_win. See below about these variables.

max: maximal size of receive buffer allowed for automatically selected receiver buffers for TCP socket. This value does not override net.core.rmem_max. Calling setsockopt() with SO_RCVBUF disables automatic tuning of that socket's receive buffer size, in which case this value is ignored.
Default: between 87380B and 4MB, depending on RAM size.

tcp_sack - BOOLEAN

Enable select acknowledgments (SACKS).

tcp_slow_start_after_idle - BOOLEAN

If set, provide RFC2861 behavior and time out the congestion window after an idle period. An idle period is defined at the current RTO. If unset, the congestion window will not be timed out after an idle period.
Default: 1

tcp_stdurg - BOOLEAN

Use the Host requirements interpretation of the TCP urgent pointer field.
Most hosts use the older BSD interpretation, so if you turn this on Linux might not communicate correctly with them.
Default: FALSE

tcp_synack_retries - INTEGER

Number of times SYNACKs for a passive TCP connection attempt will be retransmitted. Should not be higher than 255. Default value is 5, which corresponds to ~180seconds.

tcp_syncookies - BOOLEAN

Only valid when the kernel was compiled with CONFIG_SYNCOOKIES
Send out syncookies when the syn backlog queue of a socket overflows. This is to prevent against the common 'SYN flood attack'
Default: FALSE

Note, that syncookies is fallback facility.
It MUST NOT be used to help highly loaded servers to stand against legal connection rate. If you see SYN flood warnings in your logs, but investigation shows that they occur because of overload with legal connections, you should tune another parameters until this warning disappear.
See: tcp_max_syn_backlog, tcp_synack_retries, tcp_abort_on_overflow.

syncookies seriously violate TCP protocol, do not allow to use TCP extensions, can result in serious degradation of some services (f.e. SMTP relaying), visible not by you, but your clients and relays, contacting you. While you see SYN flood warnings in logs not being really flooded, your server is seriously misconfigured.

tcp_syn_retries - INTEGER

Number of times initial SYNs for an active TCP connection attempt will be retransmitted. Should not be higher than 255. Default value is 5, which corresponds to ~180seconds.

tcp_timestamps - BOOLEAN

Enable timestamps as defined in RFC1323.

tcp_tso_win_divisor - INTEGER

This allows control over what percentage of the congestion window can be consumed by a single TSO frame.
The setting of this parameter is a choice between burstiness and building larger TSO frames.
Default: 3

tcp_tw_recycle - BOOLEAN

Enable fast recycling TIME-WAIT sockets. Default value is 0.

```
        It should not be changed without advice/request of technical
        experts.

tcp_tw_reuse - BOOLEAN
        Allow to reuse TIME-WAIT sockets for new connections when it is
        safe from protocol viewpoint. Default value is 0.
        It should not be changed without advice/request of technical
        experts.

tcp_window_scaling - BOOLEAN
        Enable window scaling as defined in RFC1323.

tcp_wmem - vector of 3 INTEGERs: min, default, max
        min: Amount of memory reserved for send buffers for TCP sockets.
        Each TCP socket has rights to use it due to fact of its birth.
        Default: 4K

        default: initial size of send buffer used by TCP sockets.  This
        value overrides net.core.wmem_default used by other protocols.
        It is usually lower than net.core.wmem_default.
        Default: 16K

        max: Maximal amount of memory allowed for automatically tuned
        send buffers for TCP sockets. This value does not override
        net.core.wmem_max.  Calling setsockopt() with SO_SNDBUF disables
        automatic tuning of that socket's send buffer size, in which case
        this value is ignored.
        Default: between 64K and 4MB, depending on RAM size.

tcp_workaround_signed_windows - BOOLEAN
        If set, assume no receipt of a window scaling option means the
        remote TCP is broken and treats the window as a signed quantity.
        If unset, assume the remote TCP is not broken even if we do
        not receive a window scaling option from them.
        Default: 0

tcp_dma_copybreak - INTEGER
        Lower limit, in bytes, of the size of socket reads that will be
        offloaded to a DMA copy engine, if one is present in the system
        and CONFIG_NET_DMA is enabled.
        Default: 4096

tcp_thin_linear_timeouts - BOOLEAN
        Enable dynamic triggering of linear timeouts for thin streams.
        If set, a check is performed upon retransmission by timeout to
        determine if the stream is thin (less than 4 packets in flight).
        As long as the stream is found to be thin, up to 6 linear
        timeouts may be performed before exponential backoff mode is
        initiated. This improves retransmission latency for
        non-aggressive thin streams, often found to be time-dependent.
        For more information on thin streams, see
        Documentation/networking/tcp-thin.txt
        Default: 0

tcp_thin_dupack - BOOLEAN
        Enable dynamic triggering of retransmissions after one dupACK
```

for thin streams. If set, a check is performed upon reception of a dupACK to determine if the stream is thin (less than 4 packets in flight). As long as the stream is found to be thin, data is retransmitted on the first received dupACK. This improves retransmission latency for non-aggressive thin streams, often found to be time-dependent.
For more information on thin streams, see
Documentation/networking/tcp-thin.txt
Default: 0

UDP variables:

udp_mem - vector of 3 INTEGERs: min, pressure, max

Number of pages allowed for queueing by all UDP sockets.

min: Below this number of pages UDP is not bothered about its memory appetite. When amount of memory allocated by UDP exceeds this number, UDP starts to moderate memory usage.

pressure: This value was introduced to follow format of tcp_mem.

max: Number of pages allowed for queueing by all UDP sockets.

Default is calculated at boot time from amount of available memory.

udp_rmem_min - INTEGER

Minimal size of receive buffer used by UDP sockets in moderation. Each UDP socket is able to use the size for receiving data, even if total pages of UDP sockets exceed udp_mem pressure. The unit is byte.
Default: 4096

udp_wmem_min - INTEGER

Minimal size of send buffer used by UDP sockets in moderation. Each UDP socket is able to use the size for sending data, even if total pages of UDP sockets exceed udp_mem pressure. The unit is byte.
Default: 4096

CIPSOv4 Variables:

cipso_cache_enable - BOOLEAN

If set, enable additions to and lookups from the CIPSO label mapping cache. If unset, additions are ignored and lookups always result in a miss. However, regardless of the setting the cache is still invalidated when required when means you can safely toggle this on and off and the cache will always be "safe".
Default: 1

cipso_cache_bucket_size - INTEGER

The CIPSO label cache consists of a fixed size hash table with each hash bucket containing a number of cache entries. This variable limits the number of entries in each hash bucket; the larger the value the more CIPSO label mappings that can be cached. When the number of entries in a given hash bucket reaches this limit adding new entries causes the oldest entry in the bucket to be removed to make room.
Default: 10

cipso_rbm_optfmt - BOOLEAN

Enable the "Optimized Tag 1 Format" as defined in section 3.4.2.6 of the CIPSO draft specification (see Documentation/netlabel for details). This means that when set the CIPSO tag will be padded with empty categories in order to make the packet data 32-bit aligned.
Default: 0

cipso_rbm_structvalid - BOOLEAN

If set, do a very strict check of the CIPSO option when ip_options_compile() is called. If unset, relax the checks done during ip_options_compile(). Either way is "safe" as errors are caught else where in the CIPSO processing code but setting this to 0 (False) should result in less work (i.e. it should be faster) but could cause problems with other implementations that require strict checking.
Default: 0

IP Variables:

ip_local_port_range - 2 INTEGERS

Defines the local port range that is used by TCP and UDP to choose the local port. The first number is the first, the second the last local port number. Default value depends on amount of memory available on the system:
> 128Mb 32768-61000
< 128Mb 1024-4999 or even less.
This number defines number of active connections, which this system can issue simultaneously to systems not supporting TCP extensions (timestamps). With tcp_tw_recycle enabled (i.e. by default) range 1024-4999 is enough to issue up to 2000 connections per second to systems supporting timestamps.

ip_local_reserved_ports - list of comma separated ranges

Specify the ports which are reserved for known third-party applications. These ports will not be used by automatic port assignments (e.g. when calling connect() or bind() with port number 0). Explicit port allocation behavior is unchanged.

The format used for both input and output is a comma separated list of ranges (e.g. "1,2-4,10-10" for ports 1, 2, 3, 4 and 10). Writing to the file will clear all previously reserved ports and update the current list with the one given in the input.

Note that ip_local_port_range and ip_local_reserved_ports settings are independent and both are considered by the kernel when determining which ports are available for automatic port assignments.

You can reserve ports which are not in the current ip_local_port_range, e.g.:

```
$ cat /proc/sys/net/ipv4/ip_local_port_range
32000   61000
$ cat /proc/sys/net/ipv4/ip_local_reserved_ports
8080,9148
```

```
	although this is redundant. However such a setting is useful
	if later the port range is changed to a value that will
	include the reserved ports.

	Default: Empty

ip_nonlocal_bind - BOOLEAN
	If set, allows processes to bind() to non-local IP addresses,
	which can be quite useful - but may break some applications.
	Default: 0

ip_dynaddr - BOOLEAN
	If set non-zero, enables support for dynamic addresses.
	If set to a non-zero value larger than 1, a kernel log
	message will be printed when dynamic address rewriting
	occurs.
	Default: 0

icmp_echo_ignore_all - BOOLEAN
	If set non-zero, then the kernel will ignore all ICMP ECHO
	requests sent to it.
	Default: 0

icmp_echo_ignore_broadcasts - BOOLEAN
	If set non-zero, then the kernel will ignore all ICMP ECHO and
	TIMESTAMP requests sent to it via broadcast/multicast.
	Default: 1

icmp_ratelimit - INTEGER
	Limit the maximal rates for sending ICMP packets whose type matches
	icmp_ratemask (see below) to specific targets.
	0 to disable any limiting,
	otherwise the minimal space between responses in milliseconds.
	Default: 1000

icmp_ratemask - INTEGER
	Mask made of ICMP types for which rates are being limited.
	Significant bits: IHGFEDCBA9876543210
	Default mask:     0000001100000011000 (6168)

	Bit definitions (see include/linux/icmp.h):
		0 Echo Reply
		3 Destination Unreachable *
		4 Source Quench *
		5 Redirect
		8 Echo Request
		B Time Exceeded *
		C Parameter Problem *
		D Timestamp Request
		E Timestamp Reply
		F Info Request
		G Info Reply
		H Address Mask Request
		I Address Mask Reply

	* These are rate limited by default (see default mask above)
```

```
icmp_ignore_bogus_error_responses - BOOLEAN
	Some routers violate RFC1122 by sending bogus responses to broadcast
	frames.  Such violations are normally logged via a kernel warning.
	If this is set to TRUE, the kernel will not give such warnings, which
	will avoid log file clutter.
	Default: FALSE

icmp_errors_use_inbound_ifaddr - BOOLEAN

	If zero, icmp error messages are sent with the primary address of
	the exiting interface.

	If non-zero, the message will be sent with the primary address of
	the interface that received the packet that caused the icmp error.
	This is the behaviour network many administrators will expect from
	a router. And it can make debugging complicated network layouts
	much easier.

	Note that if no primary address exists for the interface selected,
	then the primary address of the first non-loopback interface that
	has one will be used regardless of this setting.

	Default: 0

igmp_max_memberships - INTEGER
	Change the maximum number of multicast groups we can subscribe to.
	Default: 20

conf/interface/*  changes special settings per interface (where "interface" is
		  the name of your network interface)
conf/all/*	  is special, changes the settings for all interfaces


log_martians - BOOLEAN
	Log packets with impossible addresses to kernel log.
	log_martians for the interface will be enabled if at least one of
	conf/{all,interface}/log_martians is set to TRUE,
	it will be disabled otherwise

accept_redirects - BOOLEAN
	Accept ICMP redirect messages.
	accept_redirects for the interface will be enabled if:
	- both conf/{all,interface}/accept_redirects are TRUE in the case
	  forwarding for the interface is enabled
	or
	- at least one of conf/{all,interface}/accept_redirects is TRUE in the
	  case forwarding for the interface is disabled
	accept_redirects for the interface will be disabled otherwise
	default TRUE (host)
		FALSE (router)

forwarding - BOOLEAN
	Enable IP forwarding on this interface.

mc_forwarding - BOOLEAN
```

Do multicast routing. The kernel needs to be compiled with CONFIG_MROUTE and a multicast routing daemon is required.
conf/all/mc_forwarding must also be set to TRUE to enable multicast routing for the interface

medium_id - INTEGER

Integer value used to differentiate the devices by the medium they are attached to. Two devices can have different id values when the broadcast packets are received only on one of them.
The default value 0 means that the device is the only interface to its medium, value of -1 means that medium is not known.

Currently, it is used to change the proxy_arp behavior:
the proxy_arp feature is enabled for packets forwarded between two devices attached to different media.

proxy_arp - BOOLEAN

Do proxy arp.
proxy_arp for the interface will be enabled if at least one of conf/{all,interface}/proxy_arp is set to TRUE,
it will be disabled otherwise

proxy_arp_pvlan - BOOLEAN

Private VLAN proxy arp.
Basically allow proxy arp replies back to the same interface (from which the ARP request/solicitation was received).

This is done to support (ethernet) switch features, like RFC 3069, where the individual ports are NOT allowed to communicate with each other, but they are allowed to talk to the upstream router. As described in RFC 3069, it is possible to allow these hosts to communicate through the upstream router by proxy_arp'ing. Don't need to be used together with proxy_arp.

This technology is known by different names:
  In RFC 3069 it is called VLAN Aggregation.
  Cisco and Allied Telesyn call it Private VLAN.
  Hewlett-Packard call it Source-Port filtering or port-isolation.
  Ericsson call it MAC-Forced Forwarding (RFC Draft).

shared_media - BOOLEAN

Send(router) or accept(host) RFC1620 shared media redirects.
Overrides ip_secure_redirects.
shared_media for the interface will be enabled if at least one of conf/{all,interface}/shared_media is set to TRUE,
it will be disabled otherwise
default TRUE

secure_redirects - BOOLEAN

Accept ICMP redirect messages only for gateways, listed in default gateway list.
secure_redirects for the interface will be enabled if at least one of conf/{all,interface}/secure_redirects is set to TRUE,
it will be disabled otherwise
default TRUE

```
send_redirects - BOOLEAN
	Send redirects, if router.
	send_redirects for the interface will be enabled if at least one of
	conf/{all,interface}/send_redirects is set to TRUE,
	it will be disabled otherwise
	Default: TRUE

bootp_relay - BOOLEAN
	Accept packets with source address 0.b.c.d destined
	not to this host as local ones. It is supposed, that
	BOOTP relay daemon will catch and forward such packets.
	conf/all/bootp_relay must also be set to TRUE to enable BOOTP relay
	for the interface
	default FALSE
	Not Implemented Yet.

accept_source_route - BOOLEAN
	Accept packets with SRR option.
	conf/all/accept_source_route must also be set to TRUE to accept packets
	with SRR option on the interface
	default TRUE (router)
		FALSE (host)

accept_local - BOOLEAN
	Accept packets with local source addresses. In combination with
	suitable routing, this can be used to direct packets between two
	local interfaces over the wire and have them accepted properly.
	default FALSE

rp_filter - INTEGER
	0 - No source validation.
	1 - Strict mode as defined in RFC3704 Strict Reverse Path
	    Each incoming packet is tested against the FIB and if the interface
	    is not the best reverse path the packet check will fail.
	    By default failed packets are discarded.
	2 - Loose mode as defined in RFC3704 Loose Reverse Path
	    Each incoming packet’s source address is also tested against the FIB
	    and if the source address is not reachable via any interface
	    the packet check will fail.

	Current recommended practice in RFC3704 is to enable strict mode
	to prevent IP spoofing from DDos attacks. If using asymmetric routing
	or other complicated routing, then loose mode is recommended.

	The max value from conf/{all,interface}/rp_filter is used
	when doing source validation on the {interface}.

	Default value is 0. Note that some distributions enable it
	in startup scripts.

arp_filter - BOOLEAN
	1 - Allows you to have multiple network interfaces on the same
	subnet, and have the ARPs for each interface be answered
	based on whether or not the kernel would route a packet from
	the ARP’d IP out that interface (therefore you must use source
```

based routing for this to work). In other words it allows control of which cards (usually 1) will respond to an arp request.

0 - (default) The kernel can respond to arp requests with addresses from other interfaces. This may seem wrong but it usually makes sense, because it increases the chance of successful communication. IP addresses are owned by the complete host on Linux, not by particular interfaces. Only for more complex setups like load-balancing, does this behaviour cause problems.

arp_filter for the interface will be enabled if at least one of conf/{all,interface}/arp_filter is set to TRUE,
it will be disabled otherwise

arp_announce - INTEGER

Define different restriction levels for announcing the local source IP address from IP packets in ARP requests sent on interface:
0 - (default) Use any local address, configured on any interface
1 - Try to avoid local addresses that are not in the target's subnet for this interface. This mode is useful when target hosts reachable via this interface require the source IP address in ARP requests to be part of their logical network configured on the receiving interface. When we generate the request we will check all our subnets that include the target IP and will preserve the source address if it is from such subnet. If there is no such subnet we select source address according to the rules for level 2.
2 - Always use the best local address for this target.
In this mode we ignore the source address in the IP packet and try to select local address that we prefer for talks with the target host. Such local address is selected by looking for primary IP addresses on all our subnets on the outgoing interface that include the target IP address. If no suitable local address is found we select the first local address we have on the outgoing interface or on all other interfaces, with the hope we will receive reply for our request and even sometimes no matter the source IP address we announce.

The max value from conf/{all,interface}/arp_announce is used.

Increasing the restriction level gives more chance for receiving answer from the resolved target while decreasing the level announces more valid sender's information.

arp_ignore - INTEGER

Define different modes for sending replies in response to received ARP requests that resolve local target IP addresses:
0 - (default): reply for any local target IP address, configured on any interface
1 - reply only if the target IP address is local address configured on the incoming interface
2 - reply only if the target IP address is local address configured on the incoming interface and both with the sender's IP address are part from same subnet on this interface
3 - do not reply for local addresses configured with scope host,

```
	only resolutions for global and link addresses are replied
	4-7 - reserved
	8 - do not reply for all local addresses

	The max value from conf/{all,interface}/arp_ignore is used
	when ARP request is received on the {interface}

arp_notify - BOOLEAN
	Define mode for notification of address and device changes.
	0 - (default): do nothing
	1 - Generate gratuitous arp replies when device is brought up
	    or hardware address changes.

arp_accept - BOOLEAN
	Define behavior for gratuitous ARP frames who's IP is not
	already present in the ARP table:
	0 - don't create new entries in the ARP table
	1 - create new entries in the ARP table

	Both replies and requests type gratuitous arp will trigger the
	ARP table to be updated, if this setting is on.

	If the ARP table already contains the IP address of the
	gratuitous arp frame, the arp table will be updated regardless
	if this setting is on or off.


app_solicit - INTEGER
	The maximum number of probes to send to the user space ARP daemon
	via netlink before dropping back to multicast probes (see
	mcast_solicit).  Defaults to 0.

disable_policy - BOOLEAN
	Disable IPSEC policy (SPD) for this interface

disable_xfrm - BOOLEAN
	Disable IPSEC encryption on this interface, whatever the policy



tag - INTEGER
	Allows you to write a number, which can be used as required.
	Default value is 0.

Alexey Kuznetsov.
kuznet@ms2.inr.ac.ru

Updated by:
Andi Kleen
ak@muc.de
Nicolas Delon
delon.nicolas@wanadoo.fr
```

```
/proc/sys/net/ipv6/* Variables:

IPv6 has no global variables such as tcp_*.  tcp_* settings under ipv4/ also
apply to IPv6 [XXX?].

bindv6only - BOOLEAN
        Default value for IPV6_V6ONLY socket option,
        which restricts use of the IPv6 socket to IPv6 communication
        only.
                TRUE: disable IPv4-mapped address feature
                FALSE: enable IPv4-mapped address feature

        Default: FALSE (as specified in RFC2553bis)

IPv6 Fragmentation:

ip6frag_high_thresh - INTEGER
        Maximum memory used to reassemble IPv6 fragments. When
        ip6frag_high_thresh bytes of memory is allocated for this purpose,
        the fragment handler will toss packets until ip6frag_low_thresh
        is reached.

ip6frag_low_thresh - INTEGER
        See ip6frag_high_thresh

ip6frag_time - INTEGER
        Time in seconds to keep an IPv6 fragment in memory.

ip6frag_secret_interval - INTEGER
        Regeneration interval (in seconds) of the hash secret (or lifetime
        for the hash secret) for IPv6 fragments.
        Default: 600

conf/default/*:
        Change the interface-specific default settings.


conf/all/*:
        Change all the interface-specific settings.

        [XXX:  Other special features than forwarding?]

conf/all/forwarding - BOOLEAN
        Enable global IPv6 forwarding between all interfaces.

        IPv4 and IPv6 work differently here; e.g. netfilter must be used
        to control which interfaces may forward packets and which not.

        This also sets all interfaces' Host/Router setting
        'forwarding' to the specified value.  See below for details.

        This referred to as global forwarding.

proxy_ndp - BOOLEAN
        Do proxy ndp.
```

```
conf/interface/*:
	Change special settings per interface.

	The functional behaviour for certain settings is different
	depending on whether local forwarding is enabled or not.

accept_ra - BOOLEAN
	Accept Router Advertisements; autoconfigure using them.

	Functional default: enabled if local forwarding is disabled.
	                    disabled if local forwarding is enabled.

accept_ra_defrtr - BOOLEAN
	Learn default router in Router Advertisement.

	Functional default: enabled if accept_ra is enabled.
	                    disabled if accept_ra is disabled.

accept_ra_pinfo - BOOLEAN
	Learn Prefix Information in Router Advertisement.

	Functional default: enabled if accept_ra is enabled.
	                    disabled if accept_ra is disabled.

accept_ra_rt_info_max_plen - INTEGER
	Maximum prefix length of Route Information in RA.

	Route Information w/ prefix larger than or equal to this
	variable shall be ignored.

	Functional default: 0 if accept_ra_rtr_pref is enabled.
	                    -1 if accept_ra_rtr_pref is disabled.

accept_ra_rtr_pref - BOOLEAN
	Accept Router Preference in RA.

	Functional default: enabled if accept_ra is enabled.
	                    disabled if accept_ra is disabled.

accept_redirects - BOOLEAN
	Accept Redirects.

	Functional default: enabled if local forwarding is disabled.
	                    disabled if local forwarding is enabled.

accept_source_route - INTEGER
	Accept source routing (routing extension header).

	>= 0: Accept only routing header type 2.
	< 0: Do not accept routing header.

	Default: 0

autoconf - BOOLEAN
	Autoconfigure addresses using Prefix Information in Router
	Advertisements.
```

```
        Functional default: enabled if accept_ra_pinfo is enabled.
                            disabled if accept_ra_pinfo is disabled.

dad_transmits - INTEGER
        The amount of Duplicate Address Detection probes to send.
        Default: 1

forwarding - BOOLEAN
        Configure interface-specific Host/Router behaviour.

        Note: It is recommended to have the same setting on all
        interfaces; mixed router/host scenarios are rather uncommon.

        FALSE:

        By default, Host behaviour is assumed.  This means:

        1. IsRouter flag is not set in Neighbour Advertisements.
        2. Router Solicitations are being sent when necessary.
        3. If accept_ra is TRUE (default), accept Router
           Advertisements (and do autoconfiguration).
        4. If accept_redirects is TRUE (default), accept Redirects.

        TRUE:

        If local forwarding is enabled, Router behaviour is assumed.
        This means exactly the reverse from the above:

        1. IsRouter flag is set in Neighbour Advertisements.
        2. Router Solicitations are not sent.
        3. Router Advertisements are ignored.
        4. Redirects are ignored.

        Default: FALSE if global forwarding is disabled (default),
                 otherwise TRUE.

hop_limit - INTEGER
        Default Hop Limit to set.
        Default: 64

mtu - INTEGER
        Default Maximum Transfer Unit
        Default: 1280 (IPv6 required minimum)

router_probe_interval - INTEGER
        Minimum interval (in seconds) between Router Probing described
        in RFC4191.

        Default: 60

router_solicitation_delay - INTEGER
        Number of seconds to wait after interface is brought up
        before sending Router Solicitations.
        Default: 1
```

```
router_solicitation_interval - INTEGER
	Number of seconds to wait between Router Solicitations.
	Default: 4

router_solicitations - INTEGER
	Number of Router Solicitations to send until assuming no
	routers are present.
	Default: 3

use_tempaddr - INTEGER
	Preference for Privacy Extensions (RFC3041).
	  <= 0 : disable Privacy Extensions
	  == 1 : enable Privacy Extensions, but prefer public
	         addresses over temporary addresses.
	  >  1 : enable Privacy Extensions and prefer temporary
	         addresses over public addresses.
	Default:  0 (for most devices)
		 -1 (for point-to-point devices and loopback devices)

temp_valid_lft - INTEGER
	valid lifetime (in seconds) for temporary addresses.
	Default: 604800 (7 days)

temp_prefered_lft - INTEGER
	Preferred lifetime (in seconds) for temporary addresses.
	Default: 86400 (1 day)

max_desync_factor - INTEGER
	Maximum value for DESYNC_FACTOR, which is a random value
	that ensures that clients don't synchronize with each
	other and generate new addresses at exactly the same time.
	value is in seconds.
	Default: 600

regen_max_retry - INTEGER
	Number of attempts before give up attempting to generate
	valid temporary addresses.
	Default: 5

max_addresses - INTEGER
	Maximum number of autoconfigured addresses per interface.  Setting
	to zero disables the limitation.  It is not recommended to set this
	value too large (or to zero) because it would be an easy way to
	crash the kernel by allowing too many addresses to be created.
	Default: 16

disable_ipv6 - BOOLEAN
	Disable IPv6 operation.  If accept_dad is set to 2, this value
	will be dynamically set to TRUE if DAD fails for the link-local
	address.
	Default: FALSE (enable IPv6 operation)

	When this value is changed from 1 to 0 (IPv6 is being enabled),
	it will dynamically create a link-local address on the given
	interface and start Duplicate Address Detection, if necessary.
```

When this value is changed from 0 to 1 (IPv6 is being disabled), it will dynamically delete all address on the given interface.

accept_dad - INTEGER

Whether to accept DAD (Duplicate Address Detection).
0: Disable DAD
1: Enable DAD (default)
2: Enable DAD, and disable IPv6 operation if MAC-based duplicate link-local address has been found.

force_tllao - BOOLEAN

Enable sending the target link-layer address option even when responding to a unicast neighbor solicitation.
Default: FALSE

Quoting from RFC 2461, section 4.4, Target link-layer address:

"The option MUST be included for multicast solicitations in order to avoid infinite Neighbor Solicitation "recursion" when the peer node does not have a cache entry to return a Neighbor Advertisements message. When responding to unicast solicitations, the option can be omitted since the sender of the solicitation has the correct link-layer address; otherwise it would not have be able to send the unicast solicitation in the first place. However, including the link-layer address in this case adds little overhead and eliminates a potential race condition where the sender deletes the cached link-layer address prior to receiving a response to a previous solicitation."

icmp/*:
ratelimit - INTEGER

Limit the maximal rates for sending ICMPv6 packets.
0 to disable any limiting,
otherwise the minimal space between responses in milliseconds.
Default: 1000

IPv6 Update by:
Pekka Savola <pekkas@netcore.fi>
YOSHIFUJI Hideaki / USAGI Project <yoshfuji@linux-ipv6.org>

/proc/sys/net/bridge/* Variables:

bridge-nf-call-arptables - BOOLEAN

1 : pass bridged ARP traffic to arptables' FORWARD chain.
0 : disable this.
Default: 1

bridge-nf-call-iptables - BOOLEAN

1 : pass bridged IPv4 traffic to iptables' chains.
0 : disable this.
Default: 1

bridge-nf-call-ip6tables - BOOLEAN

1 : pass bridged IPv6 traffic to ip6tables' chains.
0 : disable this.

```
	Default: 1

bridge-nf-filter-vlan-tagged - BOOLEAN
	1 : pass bridged vlan-tagged ARP/IP/IPv6 traffic to {arp,ip,ip6}tables.
	0 : disable this.
	Default: 1

bridge-nf-filter-pppoe-tagged - BOOLEAN
	1 : pass bridged pppoe-tagged IP/IPv6 traffic to {ip,ip6}tables.
	0 : disable this.
	Default: 1


proc/sys/net/sctp/* Variables:

addip_enable - BOOLEAN
	Enable or disable extension of  Dynamic Address Reconfiguration
	(ADD-IP) functionality specified in RFC5061.  This extension provides
	the ability to dynamically add and remove new addresses for the SCTP
	associations.

	1: Enable extension.

	0: Disable extension.

	Default: 0

addip_noauth_enable - BOOLEAN
	Dynamic Address Reconfiguration (ADD-IP) requires the use of
	authentication to protect the operations of adding or removing new
	addresses.  This requirement is mandated so that unauthorized hosts
	would not be able to hijack associations.  However, older
	implementations may not have implemented this requirement while
	allowing the ADD-IP extension.  For reasons of interoperability,
	we provide this variable to control the enforcement of the
	authentication requirement.

	1: Allow ADD-IP extension to be used without authentication.  This
	   should only be set in a closed environment for interoperability
	   with older implementations.

	0: Enforce the authentication requirement

	Default: 0

auth_enable - BOOLEAN
	Enable or disable Authenticated Chunks extension.  This extension
	provides the ability to send and receive authenticated chunks and is
	required for secure operation of Dynamic Address Reconfiguration
	(ADD-IP) extension.

	1: Enable this extension.
	0: Disable this extension.

	Default: 0
```

prsctp_enable - BOOLEAN

Enable or disable the Partial Reliability extension (RFC3758) which is used to notify peers that a given DATA should no longer be expected.

1: Enable extension
0: Disable

Default: 1

max_burst - INTEGER

The limit of the number of new packets that can be initially sent. It controls how bursty the generated traffic can be.

Default: 4

association_max_retrans - INTEGER

Set the maximum number for retransmissions that an association can attempt deciding that the remote end is unreachable. If this value is exceeded, the association is terminated.

Default: 10

max_init_retransmits - INTEGER

The maximum number of retransmissions of INIT and COOKIE-ECHO chunks that an association will attempt before declaring the destination unreachable and terminating.

Default: 8

path_max_retrans - INTEGER

The maximum number of retransmissions that will be attempted on a given path. Once this threshold is exceeded, the path is considered unreachable, and new traffic will use a different path when the association is multihomed.

Default: 5

rto_initial - INTEGER

The initial round trip timeout value in milliseconds that will be used in calculating round trip times. This is the initial time interval for retransmissions.

Default: 3000

rto_max - INTEGER

The maximum value (in milliseconds) of the round trip timeout. This is the largest time interval that can elapse between retransmissions.

Default: 60000

rto_min - INTEGER

The minimum value (in milliseconds) of the round trip timeout. This is the smallest time interval the can elapse between retransmissions.

Default: 1000

hb_interval - INTEGER

The interval (in milliseconds) between HEARTBEAT chunks. These chunks are sent at the specified interval on idle paths to probe the state of a given path between 2 associations.

Default: 30000

sack_timeout - INTEGER

The amount of time (in milliseconds) that the implementation will wait to send a SACK.

Default: 200

valid_cookie_life - INTEGER

The default lifetime of the SCTP cookie (in milliseconds). The cookie is used during association establishment.

Default: 60000

cookie_preserve_enable - BOOLEAN

Enable or disable the ability to extend the lifetime of the SCTP cookie that is used during the establishment phase of SCTP association

1: Enable cookie lifetime extension.
0: Disable

Default: 1

rcvbuf_policy - INTEGER

Determines if the receive buffer is attributed to the socket or to association. SCTP supports the capability to create multiple associations on a single socket. When using this capability, it is possible that a single stalled association that's buffering a lot of data may block other associations from delivering their data by consuming all of the receive buffer space. To work around this, the rcvbuf_policy could be set to attribute the receiver buffer space to each association instead of the socket. This prevents the described blocking.

1: rcvbuf space is per association
0: recbuf space is per socket

Default: 0

sndbuf_policy - INTEGER

Similar to rcvbuf_policy above, this applies to send buffer space.

1: Send buffer is tracked per association
0: Send buffer is tracked per socket.

Default: 0

sctp_mem - vector of 3 INTEGERs: min, pressure, max

Number of pages allowed for queueing by all SCTP sockets.

min: Below this number of pages SCTP is not bothered about its memory appetite. When amount of memory allocated by SCTP exceeds this number, SCTP starts to moderate memory usage.

pressure: This value was introduced to follow format of tcp_mem.

max: Number of pages allowed for queueing by all SCTP sockets.

Default is calculated at boot time from amount of available memory.

sctp_rmem - vector of 3 INTEGERs: min, default, max
See tcp_rmem for a description.

sctp_wmem - vector of 3 INTEGERs: min, default, max
See tcp_wmem for a description.

addr_scope_policy - INTEGER
Control IPv4 address scoping - draft-stewart-tsvwg-sctp-ipv4-00

0   - Disable IPv4 address scoping
1   - Enable IPv4 address scoping
2   - Follow draft but allow IPv4 private addresses
3   - Follow draft but allow IPv4 link local addresses

Default: 1

/proc/sys/net/core/*
dev_weight - INTEGER
The maximum number of packets that kernel can handle on a NAPI interrupt, it's a Per-CPU variable.

Default: 64

/proc/sys/net/unix/*
max_dgram_qlen - INTEGER
The maximum length of dgram socket receive queue

Default: 10

UNDOCUMENTED:

/proc/sys/net/irda/*
fast_poll_increase FIXME
warn_noreply_time FIXME
discovery_slots FIXME
slot_timeout FIXME
max_baud_rate FIXME
discovery_timeout FIXME
lap_keepalive_time FIXME
max_noreply_time FIXME
max_tx_data_size FIXME
max_tx_window FIXME
min_tx_turn_time FIXME