Ipsysctl tutorial 1.0.4

Oskar Andreasson

blueflux@koffein.net

Ipsysctl tutorial 1.0.4

by Oskar Andreasson

Dedications

This document is dedicated to all of you who send me reports of bugs and errors in what I have written, and to everyone else for that matter who reads this and other documentations in order to find errors and report them to the maintainers. This document is in other words dedicated to everyone who uses what others produce and release for free under the Free Licenses that are available.

I would also like to dedicate this to my family who has understood me when I didn’t deserve it, who whacked my head off for being a bum when I deserved it, and for being a generally nice bunch of people with a lot of humour.

Table of Contents

List of Tables

Preface

1. Why this document

I started writing this documentation in the hopes that it would help people understand the IP options provided by Linux 2.4, and what you can do with these options. This is a plain text documentation, hoping to give the necessary understanding and help to configure your kernel on the fly, and to get it up and running in a way that suites you. A lot of these options can also be used to increase performance, as well as strengthen the security. We will not discuss all the different sections of the sysctl in this document, but instead focus on the network sections of the system control, or sysctl as it is called. Hopefully this documentation will fill a gap in the documentation, and if you're reading this, it probably has. Mainly, there are no really good documentations detailing the whole structure of all the ipsysctl from what I have seen that documents all the networking options, except the ip-sysctl.txt file in the linux kernel documentation, which is really really brief in explaining what the options could be used for.

I believe there is a lot of documentation out there detailing the usage of these options and variables, however, none of them brings them together and describes them in detail. If you found this document to be a good read, or interesting in general, feel free to donate, help out, translate, or whatever you feel like. The main thing of all however, send bugreports so I can update the document and see to it that there is no stale errors or problems due to changes in the kernel etcetera. If you find an sysctl command that has not been documented here, or is not listed, send me a mail and I will try to get it inserted as soon as possible.

2. Intended audience & prerequisite knowledge

This document is intended for evyerone with an intermediate through advanced understanding of TCP/IP as well as the Linux operating system. You should understand TCP/IP fairly well, as well as understand what a packet header is and what parts it consists of. You will also need a lot of understanding of routing and the core of TCP/IP networking.

In general, this document was not intended for the novice Linux user, but you may have some luck checking through this document if you are experiencing specific needs. Be absolutely 100% certain that you have understood the variables in question before you do change them though, since some of them may cause really interesting results.

This document should be readable by anyone who has a interest in computers and computer networks in specific, and prerequisite knowledge. It is aimed at giving a basic understanding of the different variables available through the ipsysctl, but also to make it easier to go even further in understanding what each specific variable do.

3. How to read

This documentation can be read any way you want. If there are specific sections you are interested in, read those. If you have never set a kernel variable in linux before in your life, read the first chapter after this and then read the sections you feel intrigued by. If you feel like reading it from top to bottom, do that. If you feel like reading it backwards to find hidden messages, do that. If you feel like reading in encrypted format, well, nothing is stopping you.

What I am trying to say is the following, read the sections that you want to read and that you think would be informative to you, and if there is something you do not understand, read the corresponding bits either here or in some other document. I will not tell you how to read this document since everyone will probably have bits and pieces they already know how they work.

4. Conventions used in this document

The following conventions are used in this document when it comes to commands, files and other specific information.

- Code excerpts and commandoutputs are printed like this, with all output in fixed width font and userwritten commands in bold typeface:

  ```
  [blueflux@work1 neigh]$ ls
  default  eth0  lo
  [blueflux@work1 neigh]$
  ```

- All commands and program names in the tutorial are shown in **bold typeface**.
- All system items such as hardware, and also kernel internals or abstract system items such as the loopback interface are all shown in an italic typeface.
- computeroutput is formatted in `this way` in the text.
- filenames and paths in the filesystem are shown like `/proc/sys/net`.

5. Acknowledgements

This document is loosely based upon the documentation found in /usr/src/linux/Documentation/networking/ip-sysctl.txt. Before I say anything else, I would like to thank the people who took the time to write that document. People who read both documents will find that this document has borrowed some structure from it.

I would like to acknowledge the work done by all the people working on the networking stack and the timeconsuming job they are putting into the Linux operating system. I hope I can give something back by writing this, and other documents, to the community and by giving the coders a little relief by maintaining documentation for what they have written.

I would also like to acknowledge Fabrice Marie for the wonderful help and support with documentation standards and giving me a nudge in the right direction when there is something I did not get to work. Also, a huge thanks to all of the people at the netfilter mailing list for helping out with problems and questions, and the linuxsecurity people for listening to all of my rants and waiting for all of my excuses for not getting things published in time.

In other words, a huge thanks to everyone for helping me out when I have problems and questions.

Chapter 1. Introduction

The `/proc` filesystem is a virtual filesystem, which means that it doesn't really exist except in the "head" of the Linux kernel. The `/proc` filesystem as described here is specific to the Linux kernel, even though it may very well be present in other operating systems as well, but with a different functionality and a different meaning.

1.1. Virtual filesystems

By virtual filesystem, we simply mean that there is no trace of it on top of any of your harddrives, which all filesystems would normally leave. Nothing that you ever do to a virtual filesystem will ever do any changes to the actual harddrives themselves, but only in the primary memory. Virtual filesystems are created "on the fly" by the kernel during bootup, and it is always updated every single time you enter the filesystem, or do anything within it.

> **Note:** Virtual filesystems may take several other incarnations as well. One such is a simple RAM Harddrive, where you will be able to save files while the machine is running. On diskless Amigas this was often used to move files from one floppy to another floppy by first moving the original file to the RAM memory and then swap floppies and copy it to the new floppy. However, these files would disappear as soon as you reset or restarted the machine.

1.2. The /proc filesystem

The virtual filesystem that we call `/proc` contains loads and loads of different datastructures and information gathered from the kernel at runtime, and updated whenever you try to list or view the information. The information is all gathered and shown as a normal filesystem, and hence you can read files, traverse catalog structures, etcetera. Everything that you can do in a normal filesystem in other words. However, most of the files available through the `/proc` filesystem are only available read only, which means they can't be changed. This is because they only supply us with informational data.

One section is read-writable on the other hand, and that section is placed within the `/proc/sys`. All of the variables located in this directory and subdirectories are writable as well as readable.

This document will focus on the Internet Protocol version 4 (IPv4) section of the `/proc` located in `/proc/sys/net/ipv4` which contains all the configurable settings for the IPv4 stack, including TCP, UDP, ICMP and ARP tunable settings

1.3. A brief /proc walkthrough

The `/proc` filesystem contains a few basic directories and entries, which we will describe a little bit closer in this section before we go on to the ipv4 system.

First of all, the filesystem contains a huge set of numbered directories that come and go. Each and one of these numbered directories contains information pertaining to all of the currently active processes on the machine. When a new process is started, a new directory is created in the /proc filesystem for it, and a lot of data is created within it regarding the process, such as the commandline with which the program was started with, a link to the "current working directory", environment variables, where the executable is located, and so on.

Except this, we also have quite a few files as well as directories in the root of the /proc filesystem. This is a complete listing of them all:

```
[blueflux@work1 ]$ ls -l /proc
total 0
....
-r--r--r--    1 root     root            0 Sep 19 18:09 apm
dr-xr-xr-x    4 root     root            0 Sep 19 10:52 bus
-r--r--r--    1 root     root            0 Sep 19 18:09 cmdline
-r--r--r--    1 root     root            0 Sep 19 18:09 cpuinfo
-r--r--r--    1 root     root            0 Sep 19 18:09 devices
-r--r--r--    1 root     root            0 Sep 19 18:09 dma
dr-xr-xr-x    4 root     root            0 Sep 19 18:09 driver
-r--r--r--    1 root     root            0 Sep 19 18:09 execdomains
-r--r--r--    1 root     root            0 Sep 19 18:09 fb
-r--r--r--    1 root     root            0 Sep 19 18:09 filesystems
dr-xr-xr-x    2 root     root            0 Sep 19 18:09 fs
dr-xr-xr-x    4 root     root            0 Sep 19 18:09 ide
-r--r--r--    1 root     root            0 Sep 19 18:09 interrupts
-r--r--r--    1 root     root            0 Sep 19 18:09 iomem
-r--r--r--    1 root     root            0 Sep 19 18:09 ioports
dr-xr-xr-x   18 root     root            0 Sep 19 18:09 irq
-r--------    1 root     root     268374016 Sep 19 18:09 kcore
-r--------    1 root     root            0 Sep 19 10:52 kmsg
-r--r--r--    1 root     root            0 Sep 19 18:09 ksyms
-r--r--r--    1 root     root            0 Sep 19 18:09 loadavg
-r--r--r--    1 root     root            0 Sep 19 18:09 locks
-r--r--r--    1 root     root            0 Sep 19 18:09 mdstat
-r--r--r--    1 root     root            0 Sep 19 18:09 meminfo
-r--r--r--    1 root     root            0 Sep 19 18:09 misc
-r--r--r--    1 root     root            0 Sep 19 18:09 modules
lrwxrwxrwx    1 root     root           11 Sep 19 18:09 mounts -> self/mounts
-rw-r--r--    1 root     root          208 Sep 19 11:02 mtrr
dr-xr-xr-x    3 root     root            0 Sep 19 18:09 net
dr-xr-xr-x    2 root     root            0 Sep 19 18:09 nv
-r--r--r--    1 root     root            0 Sep 19 18:09 partitions
-r--r--r--    1 root     root            0 Sep 19 18:09 pci
dr-xr-xr-x    3 root     root            0 Sep 19 18:09 scsi
```

```
lrwxrwxrwx    1 root     root           64 Sep 19 12:01 self -> 2864
-rw-r--r--    1 root     root            0 Sep 19 18:09 slabinfo
-r--r--r--    1 root     root            0 Sep 19 18:09 stat
-r--r--r--    1 root     root            0 Sep 19 18:09 swaps
dr-xr-xr-x   10 root     root            0 Sep 19 14:39 sys
dr-xr-xr-x    2 root     root            0 Sep 19 18:09 sysvipc
dr-xr-xr-x    4 root     root            0 Sep 19 18:09 tty
-r--r--r--    1 root     root            0 Sep 19 18:09 uptime
-r--r--r--    1 root     root            0 Sep 19 18:09 version
[blueflux@work1 proc]$
```

Most of the information in the files are rather "human readable", except a few of them. However, a few of them you should not touch, such as the `kcore` file. The kcore file contains debugging information regarding the kernel, and if you try to 'cat' it, your system may very well hang up and die. If you try to copy it to a real file on the harddrive, you will very soon have filled up your whole partition, and so on. What all of this tells you is to be very careful. Mostly, none of the variables or entries in the /proc filesystem is not dangerous to watch, but a few of them are. A brief walkthrough of the most important files:

- `cmdline` - The command line issued when starting the kernel.
- `cpuinfo` - Information about the *Central Processing Unit*, who made it, known bugs, flags etcetera.
- `dma` - Contains information about all DMA channels available, and which driver is using it.
- `filesystems` - Contains short information about every single filesystem that the kernel supports.
- `interrupts` - Gives you a brief listing of all IRQ channels, how many interrupts they have seen and what driver is actually using it.
- `iomem` - A brief file containing all IO memory mappings used by different drivers.
- `ioports` - Contains a brief listing of all IO ports used by different drivers.
- `kcore` - Contains a complete memory dump. Do not cat or anything like that, you may freeze your system. Mainly used to debug the system.
- `kmsg` - Contains messages sent by kernel, is not and should not be readable by users since it may contain vital information. Main usage is to debug the system.
- `ksyms` - This contains the kernel symbol table, which is mainly used to debug the kernel.
- `loadavg` - Gives the load average of the system during the last 1, 5 and 15 minutes.
- `meminfo` - Contains information about memory usage on the system.
- `modules` - Contains information about all currently loaded modules in the kernel.
- `mounts` - Symlink to another file in the `/proc` filesystem which contains information about all mounted filesystems.
- `partitions` - Contains information about all partitions found on all drives in the system.
- `pci` - Gives tons of hardware information about all PCI devices on the system, also includes AGP devices and built in devices which are connected to the PCI bus.
- `swaps` - Contains information about all swap partitions mounted.

- `uptime` - Gives you the uptime of the computer since it was last rebooted in seconds.
- `version` - Gives the exact version string of the kernel currently running, including build date and gcc versions etcetera.

And here is a list of the main directories and what you can expect to find in there:

- `bus` - Contains information about all the buses, hardware-wise, such as USB, PCI and ISA buses.
- `ide` - Contains information about all of the IDE buses on systems that has IDE buses.
- `net` - Some basic information and statistics about the different network systems compiled into the system.
- `scsi` - This directory contains information about SCSI buses on SCSI systems.
- `sys` - Contains lots of variables that may be changed, including the `/proc/sys/net/ipv4` which will be deeply discussed in this document.

As you can see, there is literally hundreds of files in the `/proc` filesystem that may be read and checked for information, and we haven't looked at half of them here. As has already been said, we will only look closer on the ipv4 part and the variables that are tunable through the sysctl inside the `/proc` filesystem.

Chapter 2. How to set variables

The ipsysctl variables may be set in two different ways which entails two totally different methods. The first one is via the sysctl application provided with most distributions per default these days. The other way entails using the /proc filesystem, which should come with any linux installation as long as you have a kernel that has /proc filesystem turned on. In other words, any linux system you find should contain the /proc filesystem).

The sysctl command is a bit more complex than the /proc filesystem, depending on how you see things. Also, as already mentioned, if you use the sysctl application you need more than just the kernel which is almost all that is required via the /proc filesystem. One of the better things with the sysctl command is that it is much easier to maintain a larger listing of changes that we may want to do. All of the changes that we want to use on the system can then be saved into a special configuration file which contains all of the variables and their values. This way of doing things is in other words more suitable for setting variables that we want to use under all circumstances.

The /proc filesystem way of doing things is a little bit easier while tweaking around with settings. When we finally have figured out the proper setting, we may as well set it in the sysctl.conf file and see to it that sysctl is run upon boot, and we will always have our settings set to kernel. Command lines in a script which sets variables through the /proc filesystem will look much worse than sysctl commands and they are generally less readable. Therefore, if you are planning to implement a huge set of ipsysctl settings in a script or another, or if you figure out that you need to set a lot of them, then you should generally try to use the sysctl command instead.

2.1. With the sysctl application

The sysctl application can be used to either set variables through the command line, or to set a larger set of variables through a configuration file as previously described. sysctl may also set several variables through the command line at once if need be, and it may also be used to list all variables and their respective values. First of all, to list all variables possible you could issue the following command:

sysctl -a

This should list all the variables and their values separated by a "=" sign. The -a or -A sign will display all possible variables and their values. The -a option will list all variables separated from the values with a "=", while -A will show the variables and values in a table form. As of writing this, -A does not work, but should hopefully do so in the close future.

As you can see there are a lot of variables really, but most of them do not pertain to ipsysctl in specific. Also note the dotted notation of the variables. In sysctl, variables switch the "/" sign for the "." sign to separate different levels. sysctl will accept "/" instead of "." and there should be no problem really with this, but just as a note on how things look. If you would only like to read a specific variable, you would do the following:

sysctl net.ipv4.tcp_sack

If we would like to set a value with sysctl we would send the -w option to the command and then the variable we would like to write to and the new value separated by an equal sign. This would then look like this:

sysctl -w net.ipv4.tcp_sack=0

This will set the tcp_sack value to 0, then print the variable with its new value and exit. Nothing strange in other words. If we would instead like to load the configuration file as explained previously, we would run the following command:

sysctl -p

This will load all of the settings we have in the /etc/sysctl.conf file. If we would instead like to use another file than the default one, we would specify the file we would like to use after the -p option, like this:

sysctl -p /etc/testsysctl.conf

This would then load the testsysctl.conf configuration options instead of our default file. The sysctl.conf file is very basic and don’t take a lot of settings. First of all, a line starting with a ; or # is a comment as usual, and all commands starts with the path to the variable, including the variable name, and then an equal sign followed by the value to set the variable to. The path to the variable is relative to /proc/sys as with all of these settings. An example sysctl.conf file would look like this:

```
# This is a comment
net.ipv4.ip_forward = 0
net.ipv4.conf.all.rp_filter = 1
kernel.sysrq = 0
```

This file will set net.ipv4.ip_forward to 0, or in other words turn it off, which means that no IP packets will be forwarded between interfaces, if you want to share your internet connection to one or more other computers, this should be turned on. net.ipv4.conf.all.rp_filter will turn on routing policy filters. This setting tells the kernel to automatically filter packets based on their source address depending on where they come from.

Finally, kernel.sysrq does not have anything to do with networking really, it is a setting that turns off the sysrq key combination that can be used if the system has crashed. This value was added to show that there exist a lot of other settings than the ipsysctl settings in sysctl.

2.2. With /proc

The proc filesystem may very well be used to set all values in ipsysctl, however, this way of setting and reading variables should probably be more suitable for experimenting, and when we do not have access to the sysctl tool. This is also very good when we are dealing with certain variables that should not be turned on before a specific time in bootup. For example, it may be a very bad idea to turn on ip_forward before we have all the firewall rules and routes up and running.

All you need to use this method of reading and setting variables is the cat and echo commands as well as a standard shell such as bash. It is highly unlikely that you do not have any of these since all distributions carry these and should be more or less impossible to not install with the installation process.

First of all, all variables that may be used to change the default behaviour on your system resides in the /proc/sys/ directory. The settings that we are interested in during this tutorial are all placed within the /proc/sys/net/ipv4 directory. In other words, all you need to do to go there is the following command

cd /proc/sys/net/ipv4

To see all the variables available, issue the following command

ls

In other words, you should know about all of this already. If you don't, you are probably reading the wrong documentation. To see the setting in a specific variable, you would issue the **cat ip_forward** command. This would look something like this:

```
[blueflux@work1 ipv4]$ cat ip_forward
0
[blueflux@work1 ipv4]$
```

As you can see, these variables can be read by anyone who has an account on the machine in question. This could pose as a small security problem since anyone who gets on to your linux computer will be able to figure out all of your exact settings without too much hassle.

> **Caution**
>
> It is unfortunately impossible to block read access to the `/proc` filesystem as of writing this. The problem is that all read/write permissions are hardcoded within the `/proc` filesystem itself. and because of this, it is impossible to change the settings manually. If you really really need to change these settings, you can do it for the whole system from within the `linux/fs/proc` directory, which contains the source code for the Linux `/proc` filesystem.

If we would like to change the above setting we would use the echo command. The echo command will normally echo any line we provide it with back to us on the screen. However, this could be piped via pretty much any standard shell to the file that we would like to save it in. This could then look like the following in bash:

```
[root@work1 ipv4]# echo "1" > ip_forward
[root@work1 ipv4]#
```

As you can see, this time around we need to have root access to set the variable value. If we do not have root access, we would get the following error message:

```
[blueflux@work1 ipv4]$ echo "1" > ip_forward
bash: ip_forward: Permission denied
[blueflux@work1 ipv4]$
```

Do note that all the above examples takes into account that we are already within the the correct directory in the proc filesystem. This is the reason why we have not written the complete path to the variables.

Chapter 3. IPv4 variable reference

This chapter will go through each and one of the IPv4 variables possible to set via sysctl or the proc filesystem. You will be provided with a basic explanation on what behaviour the variable will change and how, as well as default behaviour, if possible, and what values the variable may be set to. We will not go into any deeper discussion about why each variable should be changed unless there are any very normal reasons to change the values. The structure used within this reference chapter will follow the same structure as the structure used within ipsysctl structure, as well as the default ipv4 directory being further structured due to its large size and mix of many different variables.

3.1. IP Variables

This list contains all of the variables available in a standard 2.4.x kernel that pertains to the IP settings. As you will see, there is a huge set of them, and some should be properly set from the beginning for you, and others may not be so properly set. Most of them should look quite proper, however, some do require some extra configuration depending on your needs, but most should be decently set for you as is.

3.1.1. ip_autoconfig

3.1.2. ip_default_ttl

The ip_default_ttl variable tells the kernel what Time To Live to set as default on packets that leaves this host. This tells how long the packets may live on the internet before they are dropped. Each time the packet passes a router, firewall, computer, etcetera, the TTL is decremented with one step.

The default value for ip_default_ttl is 64, which is a fairly good TTL which will not cause too much trouble. It is very unlikely to time out in transit to the host in question. This variable takes an unsigned integer, but the actual TTL field is only 8 bit long. The value may in other words be as high as 255 and as low as 0, however 255 could be considered rude and 0 wouldn't leave your computer at all. 64 is a good value, unless you are trying to connect to computers extremely far away counted in hops or jumps. These would then time out. As it looks today, I have pretty much never seen a host that lives more than 30 hops away on the internet, so I don't think there is any need to make this value higher than the default value for now.

Setting the TTL to 255 would be considered rude since this would make a packet live an extremely long time on the internet. If there would be a glitch in 2 routers, this packet could bounce back and forth for a huge amount of time, eating away on the bandwidth without any reason at all. Normally, don't set this value higher than 100 or something alike.

3.1.3. ip_dynaddr

The ip_dynaddr variable is used to allow a few problems with dynamic addressing to be fixed. This allows diald oneshot connections to get established by dynamically changing packet source address, and sockets if local processes. This option was implemented for TCP diald-box connections and Masquerading connections. Masquerading will in other words work 100% with this option, letting Masquerading switch source adress of packets if the boxes own address change.

This option takes an integer, but only makes use of 3 possible states, 0, 1 or 2.

- 0 means that this option is turned off, which is also the default behaviour.
- 1 means that the option is enabled and running.
- Any non 0 or 1 values means that we have turned on verbose mode, which in turn will add extra debugging messages that you may use to get things to work properly.

If this variable is turned on and forwarding interface changes, this is what may happen

- Socket and packet source address is rewritten on retransmissions while in SYN_SENT state. This is the diald-box processes.
- Outbound masqueraded source address changes on output, when internal host does retransmission, until a packet from the outside is received by the tunnel.

This is especially helpful for auto-dialup links (diald), where the actual outgoing address is unknown at the moment the link is going up. This enables the same, local and masqueraded, connection requests that brought the link up to actually establish their connections. This means that we will not have to first issue an connection request just to bring the connection up, and then have to issue the "real" connection request when we have actually established the connection.

3.1.4. ip_forward

The ip_forward variable is used to turn IP forwarding on or off. This means that we can turn off the functions for forwarding packets between interfaces, which lets the computer act as a firewall, or router. Note that this is an extremely important variable for Network Address Translation, firewalling, routing, masquerading, and all other things where we actually let packets through the box to another network, as you can understand.

This is an boolean variable. In other words, it will take a 1 or a 0. The default value for this variable is 0, or disabled. As you can understand, 0 means disabled and 1 means enabled.

Note that this is an very special variable since it will reset all configuration parameters to their default states if it is changed. For a complete list of the exact states, look closer at RFC1122 (other/rfc1122.txt) for hosts and RFC1812 (other/rfc1812.txt) for routers.

3.1.5. ip_local_port_range

The ip_local_port_range variable consists of two integers which tells the kernel which ports to use for client connections. This means, all connections going from our box to some other box and where we are the client. The first port is the lower bound and the second one is the upper bound.

The default value in this variable depends on how much memory you have. If you have more than 128 megabytes of physical memory, the lower bound will be 32768 and the upper bound will be 61000. If the computer has less than 128 megabytes of physical memory, the lower bound will be 1024 and the upper bound will be 4999, or even less.

This number defines the possible active connections which this system can issue simultaneously (ie, at the same time) to other systems that does not support the TCP extension timestamps.

If you have tcp_tw_recycle enabled (the default behaviour) range 1024-4999 is enough to issue up to 2000 connections per second to systems supporting timestamps. In other words, this should be more than enough for most of us.

3.1.6. ip_no_pmtu_disc

The ip_no_pmtu_disc disables PMTU (Path Maximum Transfer Unit) discovery if enabled. In most cases this is good, so it is per default set to FALSE (ie, Path Maximum Transfer Unit is used). However, in some cases this is bad and may lead to broken connectivity. If you are experiencing problems like this, you should turn this option off and set your MTU to a reasonable value yourself.

Do note that MTU and PMTU are two different things. MTU tells the kernel the maximum transfer unit for our connection, but not over the whole connection to the other end. PMTU discovery tries to discover the maximum transfer unit to specific hosts, including all the intermediate hops on the way there.

The default value is that the ip_no_pmtu_disc is FALSE, as already stated. If this is set to TRUE, PMTU discovery is turned off. The ip_no_pmtu_disc takes a boolean value, in other words either an 1 or a 0, where 1 is on and 0 is off.

3.1.7. ip_nonlocal_bind

The ip_nonlocal_bind variable allows us to set if local processes should be able to bind to non-local IP addresses. This could be quite useful, in such cases where we want specific programs or applications to be able to listen to non-local IP adresses, such as sniffing for traffic to a specific host which may commit bad things, etcetera. The variable may, however, break some applications and they will no longer work.

The ip_nonlocal_bind variable takes a boolean value which can be set to 1 or 0. If the variable is set to 0, this option is turned off and if it is set to 1 it is turned on. The default value is to turn this option off, or 0 in other words.

3.1.8. ipfrag_high_thresh

The ipfrag_high_thresh tells the kernel the maximum amount of memory to use to reassemble IP fragments. When and if the high threshold is reached, the fragment handler will toss all packets until the memory usage reaches ipfrag_low_thresh instead. This means that all fragments that reached us during this time will have to be retransmitted.

Packets are fragmented if they are too large to pass through a certain pipe. If they are to large, the box that is trying to transmit them breaks them down into smaller pieces and send each piece one by one. When these fragments reaches their destination, they need to be defragmented (ie, put together again) to be read properly. Note that IP Fragmentation are in general a good thing, but there are a lot of people that do bad things with them since fragments are inherently a security problem.

The ipfrag_high_thresh variable takes an integer value, which would mean 0 through 2147483647 bytes can be assigned to be the upper limit of this function. The default value is 262144 bytes, or 256 kilobytes, which should work well in even the most extreme cases.

3.1.9. ipfrag_low_thresh

This option has a lot to do with the ipfrag_high_thresh option. The ipfrag_low_thresh is the lower limit at which packets should start being assembled again. What this means, all in all, is that our fragmentation handler has an queue that grows larger the more packets are waiting in the queue to be defragmentized, when this queue grows to ipfrag_high_thresh byte size, the fragmentation handler queue will stop queueing any further fragments until we reach the ipfrag_low_thresh again. This stops our system from being overloaded with fragmentized packets and may stop certain Denial of Service attacks.

This variable takes an integer value between 0 and 2147483647, and refers to the amount of bytes used at which the fragmentation handler should resume the receiving of IP fragments again. Per default it is set to 196608 bytes, or 192 kilobytes which should be a reasonable amount of memory set aside for this task even in the hardest of attacks. This value should be lower than ipfrag_high_thresh, or else it will be invalid.

3.1.10. ipfrag_time

The ipfrag_time variable tells the IP fragmentation handler how long to keep an IP fragment in memory, counted in seconds. This only refers to fragments that has been impossible to reassemble since fragments that has been assembled most probably has already been sent on to either the next layer, or to the next host.

The ipfrag_time variable takes an integer as its input and the value is counted as seconds. In other words, if you input 5 to this variable, it counts as 5 seconds.

3.2. Inet peer storage

The inet peer storage contains information pertaining to specific peers, or nodes on the Internet. However, it only contains information with a long life expectancy, and information that is not dependant upon routes. For the moment, this means that it only contains information about the ID field for the next outgoing packet. There are a few variables that changes the behaviour of the inet peer storage today, mainly how often garbage collecting is done, as well as how long time to live each peer has in the storage.

3.2.1. inet_peer_gc_maxtime

The `inet_peer_gc_maxtime` variable tells the garbage collector how often to pass over the inet peer storage memory pool during low, or absent, memory pressure. This value is in effect under the reversed conditions of the inet_peer_gc_mintime in other words. It works exactly the same as the inet_peer_gc_mintime, except for the fact that it will be in effect under different system loads. This variable is also measured in jiffies, which is explained closer in appendix A.

The inet_peer_gc_maxtime variable takes an integer value and has a default value of 120 jiffies. 120 jiffies should be a good value for most workstations and servers.

3.2.2. inet_peer_gc_mintime

The inet_peer_gc_mintime variable sets the minimum time between garbage collections (gc) passes in the inet peer storage under heavy memory pressure. If the system is under heavy utilization and there is a lot of constraints on the memory pool, this timer is used to tell the garbage collector how often to pass over the memory pool used by the inet peer storage, in jiffies. For a complete explanation of jiffies, see appendix A.

The inet_peer_gc_mintime variable takes an integer value and has a default value of 10 jiffies. This should be a fairly good value for most users and servers.

3.2.3. inet_peer_maxttl

This is the maximum time to live for the inet peer entries. Unused entries will expire after this period of time if there is no memory pressure on the pool. This would in other words mean when the number of entries in the pool is very small, and likely situations.

The inet_peer_maxttl variable takes an integer value, and is measured in jiffies. For a complete explanation of jiffies, see appendix A.

3.2.4. inet_peer_minttl

This is the minimum time to live for inet peer entries. This should be set to an high enough value to cover fragment time to live in the reassembling side of fragmented packets. The minimum time to live is guaranteed if the pool size is less than inet_peer_threshold.

The inet_peer_minttl variable takes an integer value, and is measured in jiffies. For a complete explanation of jiffies, see appendix A.

3.2.5. inet_peer_threshold

The inet_peer_threshold variable tells the approximate size of the inet peer storage. When this limit is reached, peer entries will be thrown away agressively, using the inet_peer_gc_mintime timeout. This threshold will also determine how long an entry may "live" in the peer storage, in other word it is one of the parts which decides the entries time to live. To put it simple, the higher this value is, the longer the time to live within your system.

This variable takes an integer and defaults to the value 65664 bytes.

3.3. TCP Variables

This section will take a brief look at the variables that changes the behaviour of the TCP variables. These variables are normally set to a pretty good value per default and most of them should never ever be touched, except when asked by authoritative developers! They are mainly described here, only for those who are curious about their basic meaning.

3.3.1. tcp_abort_on_overflow

The tcp_abort_on_overflow variable tells the kernel to reset new connections if the system is currently overflowed with new connection attempts that the daemon(s) can not handle. What this means, is that if the system is overflowed with 1000 large requests in a burst, connections may be reset since we can not handle them if this variable is turned on. If it is not set, the system will try to recover and handle all requests.

This variable takes an boolean value (ie, 1 or 0) and is per default set to 0 or FALSE. Avoid enabling this option except as a last resort since it most definitely harm your clients. Before considering using this variable you should try to tune up your daemons to accept connections faster.

3.3.2. tcp_adv_win_scale

This variable is used to tell the kernel how much of the socket buffer space should be used for TCP window size, and how much to save for an application buffer. If tcp_adv_win_scale is negative, the following equation is used to calculate the buffer overhead for window scaling:

$$\text{bytes} - \frac{\text{bytes}}{2^{(-\text{tcp\_adv\_win\_scale}}}$$

Where bytes are the amount of bytes in the window. If the tcp_adv_win_scale value is positive, the following equation is used to calculate the buffer overhead:

$$\frac{\text{bytes}}{2^{\text{tcp\_adv\_win\_scale}}}$$

The tcp_adv_win_scale variable takes an integer value and is per default set to 2. This in turn means that the application buffer is 1/4th of the total buffer space specified in the tcp_rmem variable.

3.3.3. tcp_app_win

This variable tells the kernel how many bytes to reserve for a specific TCP window in the TCP sockets memory buffer where the specific TCP window is transfered in. This value is used in a calculation that specifies how much of the buffer space to reserve that looks as the following:

$$\frac{\text{window}}{2^{\text{tcp\_app\_win}}}$$

As you may understand from the above calculation, the larger this value gets, the smaller will the buffer space be for the specific window. The only exception to this calculation is 0, which tells the kernel to reserve no space for this specific connection. The default value for this variable is 31 and should in general be a good value. Do not change this value unless you know what you are doing.

3.3.4. tcp_dsack

This option is required to send duplicate SACKs which was briefly described in the tcp_sack variable explanation. This is described in detail within the RFC 2883. This RFC document explains in detail how to handle situations where a packet is received twice or out of order. D-SACK is an extension to standard SACK and is used to tell the sender when a packet was received twice (ie, it was duplicated). The D-SACK data can then be used by the transmitter to improve network settings and so on. This should be 100% backwards compatible with older implementations as long as the previous implementors have not tried to implement this into the old SACK option in their own fashion. This is extremely rare and should not be a problem for anyone.

The tcp_dsack variable uses a boolean value and is per default set to 1, or turned on. Of course, this behaviour is *only* used if tcp_sack is turned on since tcp_dsack is heavily dependant upon tcp_sack. In almost all cases this should be a good idea to have turned on.

3.3.5. tcp_ecn

The tcp_ecn variable turns on Explicit Congestion Notification in TCP connections. This is used to automatically tell the host when there are congestions in a route to a specific host or a network. This can be used to throttle the transmitters to send packets in a slower rate over that specific router or firewall. Explicit Congestion Notification (ECN) is explained in detail in the *RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP* document and there is also a performance evaluation of the addition of ECN available in the *RFC 2884 - Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks* document.

Briefly, this document details how we could notify other hosts when we are congested or not, which in turn will make us able to choose other routes in preference over the currently used route, or to simply send less data until we no longer receive congestion messages.

> **Caution**
>
> There are still some old firewalls and routers out on the Internet that will filter away all IP packets that has the ECN bits set. They are fairly uncommon these days, but if you are unlucky, you may run into them. If you do experience connection problems to specific hosts, try turning ECN off and see how things go. If you find the actual host blocking the ECN packets, try getting in touch with the administrators and warn them about this. A deeper explanation of the problem, as well as a list of the most common hardware that causes this trouble is available, and can be found in the *Other resources* appendix, under the *ECN-under-Linux Unofficial Vendor Support Page* heading.

The tcp_ecn variable takes a boolean value and is per default set to 0, or turned off. If you want to turn this on in your kernel, you should set this variable to 1.

3.3.6. tcp_fack

The tcp_fack variable enables the Forward Acknowledgement system in Linux. Forward Acknowledgement is a special algorithm that works on top of the SACK options, and is geared at congestion controlling.

The main idea of FACK algorithm is to consider the most forward selective acknowledgement sequence number as a sign that all the previous un(selectively) acknowledged segments were lost. This observation allows to improve recovery of losses singificantly. This assumption breaks in presence of packet reordering, in which case the FACK algorithm is automatically turned off for that specific connection.

This algorithm was originally created by Matthew Mathis and co-authors. You can find the papers describing the algorithm more closely over at http://www.psc.edu/~mathis/.

The tcp_fack variable takes a boolean value, and is per default set to 1, or turned on. This behaviour is not used if tcp_sack is turned off since it is heavily dependant upon tcp_sack.

3.3.7. tcp_fin_timeout

The tcp_fin_timeout variable tells kernel how long to keep sockets in the state FIN-WAIT-2 if you were the one closing the socket. This is used if the other peer is broken for some reason and don't close its side, or the other peer may even crash unexpectedly. Each socket left in memory takes approximately 1.5Kb of memory, and hence this may eat a lot of memory if you have a moderate webserver or something alike.

This value takes an integer value which is per default set to 60 seconds. This used to be 180 seconds in 2.2 kernels, but was reduced due to the problems mentioned above with webservers and problems that arose from getting huge amounts of connections.

Also see the tcp_max_orphans and tcp_orphan_retries variables for more information.

3.3.8. tcp_keepalive_intvl

The tcp_keepalive_intvl variable tells the kernel how long to wait for a reply on each keepalive probe. This value is in other words extremely important when you try to calculate how long time will go before your connection will die a keepalive death.

The variable takes an integer value and the default value is 75 seconds. This is in the higher regions and should be concidered the higher threshold on what values should be concidered normal to use. The default values of the tcp_keepalive_probes and tcp_keepalive_intvl can be used to get the default time it will take before the connection is timed out because of keepalive.

With the default values of sending 9 probes with 75 seconds for each, it would take approximately 11 minutes before the connection is timed out, counting from when we start the probing which in turn will happen 2 hours from the time we last saw any traffic on the connection.

3.3.9. tcp_keepalive_probes

The tcp_keepalive_probes variable tells the kernel how many TCP keepalive probes to send out before it decides a specific connection is broken.

This variable takes an integer value, which should generally not be set higher than 50 depending on your tcp_keepalive_time value and the tcp_keepalive_interval. The default value is to send out 9 probes before telling the application that the connection is broken.

3.3.10. tcp_keepalive_time

The tcp_keepalive_time variable tells the TCP/IP stack how often to send TCP keepalive packets to keep an connection alive if it is currently unused. This value is only used when keepalive is enabled.

The tcp_keepalive_time variable takes an integer value which is counted in seconds. The default value is 7200 seconds, or 2 hours. This should be a good value for most hosts and will not take too much network resources from you. Do not set this value to low since it will then use up your network resources with unnecessary traffic.

3.3.11. tcp_max_orphans

The tcp_max_orphans variable tells the kernel how many TCP sockets that are not attached to any user file handle to maintain. In case this number is exceeded, orphaned connections are immediately reset and a warning is printed.

The only reason for this limit to exist is to prevent some simple DoS attacks. Generally you should not rely on this limit, nor should you lower it artificially. If need be, you should instead increase this limit if your network environment requires such an update. Increasing this limit may require that you get more memory installed to your system. If you hit this limit, you may also tune your network services a little bit to linger and kill sockets in this state more aggressively.

This variable takes an integer value and is per default set to 8192, but heavily depends upon how much memory you have. Each orphan that currently lives eats up 64Kb of unswappable memory, which means that one hell of a lot of data will be used up if problems arise.

> **Note:** If you run into this limit, you will get an error message via the syslog facility kern.info that looks something like this:
>
> ```
> TCP: too many of orphaned sockets
> ```
>
> If this shows up, either upgrade the box in question or look closer at the tcp_fin_timeout or tcp_orphans_retries which should give you some help with getting rid of huge amounts of orphaned sockets.

3.3.12. tcp_max_syn_backlog

The tcp_max_syn_backlog variable tells your box how many SYN requests to keep in memory that we have yet to get the third packet in a 3-way handshake from. The tcp_max_syn_backlog variable is overridden by the tcp_syncookies variable, which needs to be turned on for this variable to have any effect. If the server suffers from overloads at peak times, you may want to increase this value a little bit.

This variable takes an integer value and is per default set to different values depending on how much memory you have. If you have less than 128 Mb of RAM, it is set to a maximum of 128 SYN backlog requests. If you have more than 128 Mb of RAM, it is set to 1024 SYN backlog requests.

Caution

If this value is raised to a larger value than 1024 it would most probably be better to change the TCP_SYNQ_HSIZE value and recompile your kernel. The TCP_SYNQ_HSIZE variable is set in linux/include/tcp.h. This value should be set so to keep this formula true:

TCP_SYNQ_HSIZE*16<=tcp_max_syn_backlog

In other words, TCP_SYNQ_HSIZE times 16 should be smaller than or equal to tcp_max_syn_backlog.

3.3.13. tcp_max_tw_buckets

The tcp_max_tw_buckets variable tells the system the maximum number of sockets in TIME-WAIT to be held simultaneously. If this number is exceeded, the exceeding sockets are destroyed and a warning message is printed to you. The reason for this limit to exist is to get rid of really simple DoS attacks.

The tcp_max_tw_buckets variable takes an integer value which tells the system at which point to start destroying timewait sockets. The default value is set to 180000. This may sound much, but it is not. If anything, you should possibly need to increase this value if you start receiving errors due to this setting.

Caution

You should not lower this limit artificially. If you start receiving errors indicating this problem in normal operation, you should instead increase this value if your network requires so. This may lead to the requirement of more memory installed in the machine in question.

3.3.14. tcp_mem

The tcp_mem variable defines how the TCP stack should behave when it comes to memory usage. It consists of three values, just as the tcp_wmem and tcp_rmem variables. The values are measured in memory pages (in short, pages). The size of each memory page differs depending on hardware and configuration options in the kernel, but on standard i386 computers, this is 4 kilobyte or 4096 bytes. On some newer hardware, this is set to 16, 32 or even 64 kilobytes. All of these values have no real default value since it is calculated at boottime by the kernel, and should in most cases be good for you and most usages you may encounter.

The first value specified in the tcp_mem variable tells the kernel the low threshold. Below this point, the TCP stack do not bother at all about putting any pressure on the memory usage by different TCP sockets.

The second value tells the kernel at which point to start pressuring memory usage down. This so called memory pressure mode is continued until the memory usage enters the lower threshold again, and at which point it enters the default behaviour of the low threshold again. The memory pressure mode presses down the TCP receive and send buffers for all the sockets as much as possible, until the low mark is reached again.

The final value tells the kernel how many memory pages it may use maximally. If this value is reached, TCP streams and packets start getting dropped until we reach a lower memory usage again. This value includes all TCP sockets currently in use.

> **Tip:** This variable may give tremenduous increase in throughput on high bandwidth networks, if used properly together with the tcp_rmem and tcp_wmem variable. The tcp_rmem variable doesn't need too much manual tuning however, since the Linux 2.4 kernels has very good autotuning handlings on this aspect, but the other two may be worth looking at. For more information about this, look at the *TCP Tuning Guide*.

3.3.15. tcp_orphan_retries

The tcp_orphan_retries variable tells the TCP/IP stack how many times to retry to kill connections on the other side before killing it on our own side. If your machine runs as a highly loaded http server it may be worth thinking about lowering this value. http sockets will consume large amounts of resources if not checked.

This variable takes an integer value. The default value for this variable is 7, which would approximately correspond to 50 seconds through 16 minutes depending on the Retransmission Timeout (RTO). For a complete explanation of the RTO, read the "3.7. Data Communication" section in RFC 793 - Transmission Control Protocol.

Also see the tcp_max_orphans variable for more information.

3.3.16. tcp_reordering

The tcp_reordering variable tells the kernel how much a TCP packet may be reordered in a stream without assuming that the packet was lost somewhere on the way. If the packet is assumed lost, the TCP stack will automatically go back into a slow start since it believes packets may have been lost due to congestion somewhere. The TCP stack will also fall back from using the FACK algorithm for this specific host in the future.

This variable takes an integer variable and is per default set to 3. This should in general be a good value and you should not touch it. If this value is lowered, it may result in bad network performance, especially if packets often get reordered in connections.

> **Note:** This variable is overridden by the **reordering** option in the **ip route** command starting with kernels 2.3.15 and higher. If **reordering** is not given to the **ip route** command, the default is taken from the sysctl tcp_reordering.

3.3.17. tcp_retrans_collapse

This variable implements a bug in the TCP protocol so it will be able to talk to certain other buggy TCP stacks. Without implementing this bug in the TCP stack, we would be unable to talk to certain printers that has this bug built in. This bug makes the TCP stack try to send bigger packets on retransmission of packets to work around bugs in those printers and other hardware implementations.

This variable takes a boolean value and is normally set to 1, or on. Implementing this bug workaround will not break compatibility from our host to others, but it will make it possible to speak to those printers. In general, it should not be a dangerous workaround, but you may turn it off if you receive weird error messages.

3.3.18. tcp_retries1

The tcp_retries1 variable tells the kernel how many times it should retry to get to a host before reaching a decision that something is wrong and that it should report the suspected problem to the network layer. The minimal value here specified by RFC ???? is 3, which is also the default. This corresponds to 3 seconds through 8 minutes depending on your Retransmission timeout (RTO). For a good explanation of the Retransmission timeout, read the "3.7. Data Communication" section in RFC 793 - Transmission Control Protocol.

This variable takes an integer, which is per default set to 3 as explained above. The lower limit is 3 if you want to follow standards, and the upper bound should be lower than 100 or so since timeouts could be worse than horrible if this high.

3.3.19. tcp_retries2

The tcp_retries2 value tells the kernel how many times to retry before killing an alive TCP connection. This limit is specified to a minimum of 100 seconds in RFC 1122, but is normally way to short.

The variable takes an integer value and is set to 15 per default. This value corresponds to 13-30 minutes depending on the Retransmission timeout (RTO). Generally this should be a good timeout, you may bring it down but not necessarily.

3.3.20. tcp_rfc1337

The tcp_rfc1337 variable implements the solution found in *RFC 1337 - TIME-WAIT Assassination Hazards in TCP* to TIME-WAIT Assassination. In short, the problem is that old duplicate packets may interfer with new connections, and lead to three different problems. The first one is that old duplicate data may be accepted erroneously in new connections, leading to the sent data becoming corrupt. The second problem is that connections may become desynchronized and get into an ACK loop because of old duplicate packets entering new connections, which will become desynchronized. The third and last problem is that old duplicate packets may enter newly established connections erroneously and kill the new connection.

There are three possible solutions to this according to the mentioned RFC, however, one solution is only partial and not a long term solution, while the last requires heavy modifications of the TCP protocol, and is hence not a viable option.

The final solution that the linux kernel implements with this option, is to simply ignore RST packets sent to a socket while it is in the TIME-WAIT state. In use together with 2 minute Maximum Segment Life (MSL), this should eliminate all three problems discussed in RFC 1337.

3.3.21. tcp_rmem

The tcp_rmem variable is pretty much the same as the tcp_wmem, except in one large area. It tells the kernel the TCP receive memory buffers instead of the transmit buffer which is defined in tcp_wmem. This variable takes 3 different values, just the same as the tcp_wmem variable.

The first value tells the kernel the minimum receive buffer for each TCP connection, and this buffer is always allocated to a TCP socket, even under high pressure on the system. This value is set to 4096 bytes, or 4 kilobytes, in newer kernels, but was in previous kernels set to 8192 bytes or 8 kilobytes. This should generally be a good value, and you should avoid raising this value if you are sporadically experiencing large bursts and high network loads since the system may get into even worse problems then.

The second value specified tells the kernel the default receive buffer allocated for each TCP socket. This value overrides the /proc/sys/net/core/rmem_default value used by other protocols. The default value here is 87380 bytes, or 85 kilobytes. This value is used together with tcp_adv_win_scale and tcp_app_win to calculate the TCP window size, which is discussed within the explanations of those variables. This value should under normal circumstances not be touched either since it may result in similar problems as with the first value in this variable.

> **Tip:** This variable may give tremenduous increase in throughput on high bandwidth networks, if used properly together with the tcp_mem and tcp_wmem variable. The tcp_rmem variable doesn't need too much manual tuning however, since the Linux 2.4 kernels has very good autotuning handlings on this aspect, but the other two may be worth looking at. For more information about this, look at the *TCP Tuning Guide*.

The third and last value specified in this variable specifies the maximum receive buffer that can be allocated for a TCP socket. This value is overridden by the /proc/sys/net/core/rmem_max if the ipv4 value is larger than the core value. You need to look at the core value before you do any changes to the ipv4 value in other words. The default value here is a double up of the second value specified. In other words, 87380 * 2 bytes, or 174760 bytes (170 kilobytes). Generally this should be a good value and should not need to be changed.

3.3.22. tcp_sack

The tcp_sack variable enables Selective Acknowledgements (SACK) as they are defined in *RFC 2883 - An Extension to Selective Acknowledgement (SACK) Option for TCP* and *RFC 2883 - An Extension to Selective Acknowledgement (SACK) Option for TCP*. These RFC documents contain information on an TCP option that was especially developed to handle lossy connections.

If this variable is turned on, our host will set the SACK option in the TCP option field in the TCP header when it sends out a SYN packet. This tells the server we are connecting to that we are able to handle SACK. In the future, if the server knows how to handle SACK, it will then send ACK packets with the SACK option turned on. This option selectively acknowledges each segment in a TCP window. This is especially good on very lossy connections (connections that loose a lot of data in the transfer) since this makes it possible to only retransmit specific parts of the TCP window which lost data and not the whole TCP window as the old standards told us to do. This means that if a certain segment of a TCP window is not received, the receiver will not return a SACK for that segment. The sender will then know which packets where not received by the receiver, and will hence retransmit that packet. For redundancy, this option will fill up all space possibly within the option space, 40 bytes per segment. Each SACK'ed packet takes up 2 32-bit unsigned integers and hence the option space can contain 4 SACK'ed segments. However, normally the timestamp option is used in conjunction with this option. The timestamp option takes up 10 bytes of data, and hence only 3 segments may be SACK'ed in each packet in normal operation.

If you know that you will be sending data over an extremely lossy connection such as a bad internet connection at one point or another, this variable is recommended to turn on. However, if you will only send data over an internal network consisting of a perfect condition 2 feet cat-5 cable and both machines being able to keep up with maximum speed without any problems, you should not need it. This option is not required, but it is definitely a good idea to have it turned on. Note that the SACK option is 100% backwards compatible, so you should not run into any problems talking to any other hosts on the internet who do not support it.

The tcp_sack option takes a boolean value. This is per default set to 1, or turned on. This is generally a good idea and should cause no problems.

3.3.23. tcp_stdurg

This variable enables or disables RFC 1122 compliance. The default behaviour is to be BSD 4.2 compliant, which follows the RFC 793 explanation of the URG flag. If this variable is turned on, we may be unable to communicate properly with certain hosts on the internet, or more specifically, those hosts on the internet that are BSD 4.2 compliant. For more information on the changes, read the *RFC 1122 - Requirements for Internet Hosts -- Communication Layers* under the section "4.2.2.4 Urgent Pointer: RFC 793 Section 3.1 explanation" which refers back to *RFC 793 - Transmission Control Protocol* as can be seen in the name of the section mentioned.

The tcp_stdurg variable takes a boolean value and is per default set to 0, or FALSE. If this is turned on, your box may be unable to talk to certain hosts as described above.

3.3.24. tcp_syn_retries

The tcp_syn_retries variable tells the kernel how many times to try to retransmit the initial SYN packet for an active TCP connection attempt.

This variable takes an integer value, but should not be set higher than 255 since each retransmission will consume huge amounts of time as well as some amounts of bandwidth. Each connection retransmission takes aproximately 30-40 seconds. The default setting is 5, which would lead to an aproximate of 180 seconds delay before the connection times out.

3.3.25. tcp_synack_retries

The tcp_synack_retries setting tells the kernel how many times to retransmit the SYN,ACK reply to an SYN request. In other words, this tells the system how many times to try to establish a passive TCP connection that was started by another host.

This variable takes an integer value, but should under no circumstances be larger than 255 for the same reasons as for the tcp_syn_retries variable. Each retransmission will take aproximately 30-40 seconds. The default value of the tcp_synack_retries variable is 5, and hence the default timeout of passive TCP connections is aproximately 180 seconds.

3.3.26. tcp_syncookies

The tcp_syncookies variable is used to send out so called syncookies to hosts when the kernels syn backlog queue for a specific socket is overflowed. This means that if our host is flooded with several SYN packets from different hosts, the syn backlog queue may overflow, and hence this function starts sending out cookies to see if the SYN packets are really legit.

This variable is used to prevent an extremely common attack that is called a "syn flood attack". The tcp_syncookies variable takes an boolean value which can either be set to 0 or 1, where 0 means off. The default setting is to turn this function off.

Caution

There has been a lot of discussions about the problems and flaws with syncookies in the past. Personally, I choose to look on SYN cookies as something fairly usefull, and since it is not causing any strangeness under normal operation, it should not be very dangerous. However, it may be dangerous, and you may want to see below.

The tcp_syncookies option means that under high load the system will make new connections without advanced features like ECN or SACK being used. If syncookies are being triggered during normal load rather than an attack you should tune the tcp queue length and the servers handling the load.

You must not use this facility to help a highly loaded server to stand down from legal connections. If you start to see syn flood warnings in your logs, and they show out to be legit connections, you may tune the tcp_max_syn_backlog, tcp_synack_retries and tcp_abort_on_overflow variables.

3.3.27. tcp_timestamps

The tcp_timestamps variable tells the kernel to use timestamps as defined in RFC 1323. In short, this is an TCP option that can be used to calculate the Round Trip Measurement in a better way than the retransmission timeout method can. This should be backwards compatible in almost all circumstances so you could very well turn this on if your host lives on a high speed network. If you only use up to an 10mbps connection of some sort(LAN or Internet or anything for that matter), you should manage fairly well without this option, and at really low speeds, you may even be better off with this variable turned off.

This variable takes a boolean value and is per default set to 1, or enabled. Generally this should be a good idea to have turned on. The only exception would be if you live on an extremely slow connection such as a 56 kbps modem connection to the Internet.

For more technical information about this option read section 4 of the *RFC 1323 - TCP Extensions for High Performance*. This document discusses the technical and theoretical introduction of these options and how it should work.

3.3.28. tcp_tw_recycle

This variable enables the fast recycling function of TIME-WAIT sockets. Unless you know what you are doing you should not touch this function at all.

The tcp_tw_recycle variable takes an integer value and the default value is 0 from my experience and my understanding of the source code of linux. In other words, the statement in the linux/Documentation/ip-sysctl.txt file is wrong unless I am mistaken.

> **Caution**
>
> Do not reset this from its default value unless you know what you are doing and/or have gotten the advice or request from an technical expert or kernel coder.

3.3.29. tcp_window_scaling

The tcp_window_scaling variable enables window scaling as it is defined in RFC 1323. This RFC specifies how we can scale TCP windows if we are sending them over Large Fat Pipes (LFP). When sending TCP packets over these large pipes, we experience heavy bandwidth loss due to the channels not being fully filled while waiting for ACK's for our previous TCP windows. The main problem is that a TCP Window can not be larger than 2**16 bytes, or 65Kb large. Enabling tcp_window_scaling enables a special TCP option which makes it possible to scale these windows to a larger size, and hence reduces bandwidth losses due to not utilizing the whole connection.

This variable takes a boolean value and is per default set to 1, or true. If you want to turn this off, set it to 0.

For more information about TCP window scaling, read the *RFC 1323 - TCP Extensions for High Performance*.

3.3.30. tcp_wmem

This variable takes 3 different values which holds information on how much TCP sendbuffer memory space each TCP socket has to use. Every TCP socket has this much buffer space to use before the buffer is filled up. Each of the three values are used under different conditions.

The first value in this variable tells the minimum TCP send buffer space available for a single TCP socket. This space is always allocated for a specific TCP socket opened by a program as soon as it is opened. This value is normally set to 4096 bytes, or 4 kilobytes.

The second value in the variable tells us the default buffer space allowed for a single TCP socket to use. If the buffer tries to grow larger than this, it may get hampered if the system is currently under heavy load and don't have a lot of memory space available. It may even have to drop packets if the system is so heavily loaded that it can not give more memory than this limit. The default value set here is 16384 bytes, or 16 kilobytes of memory. It is not very wise to raise this value since the system is most probably already under heavy memory load and usage, and this would hence lead to even more problems for the rest of the system. This value overrides the /proc/sys/net/core/wmem_default value that is used by other protocols, and is usually set to a lower value than the core value.

The third value tells the kernel the maximum TCP send buffer space. This defines the maximum amount of memory a single TCP socket may use. Per default this value is set to 131072, or 128 kilobytes. This should be a reasonable value for most circumstances, and you will most probably never need to change these values. However, if you ever do need to change it, you should keep in mind that the /proc/sys/net/core/wmem_max value overrides this value, and hence this value should always be smaller than that value.

> **Tip:** This variable may give tremenduous increase in throughput on high bandwidth networks, if used properly together with the tcp_mem and tcp_rmem variable. The tcp_wmem variable is the variable of the three which may give the most gain from this kind of tweaking. Do note that you will see almost no gain on slower networks than giga ethernet networks. For more information about this, look at the *TCP Tuning Guide*.

3.4. ICMP Variables

These are the variables available in ipsysctl to change the behaviour for ICMP traffic. These are rather simple and should not pose any real problem to understand as some of the TCP variables may. They are generally very simple and mainly tell the kernel how to react on different ICMP types and if they are to be limited etcetera.

3.4.1. icmp_echo_ignore_all

If this value is set to 1, in other words on or true, the kernel chooses to totally ignore all ICMP Echo requests. This variable takes a boolean value and is per default set to false, or off. If this is variable is turned on, you and others will be unable to ping the machine in question which is generally a bad thing. Of course, everyone has different opinions about this, some say it is good because people will be unable to ping you and hence know you are there, some say it is bad because you want people to know you are available on the internet. A lot of tools and applications rely upon ICMP Echo requests, some good, some bad as always.

3.4.2. icmp_echo_ignore_broadcasts

This variable works precisely the same as icmp_echo_ignore_all except that it will only ignore those ICMP messages sent to broadcast or multicast addresses. It should be quite obvious why this is good, it would among other things stop this specific host from being part of smurf attacks and likely problems. Broadcast pings are generally bad unless you are using this to find out how many hosts on your network(s) are up or not.

The icmp_echo_ignore_broadcasts variable takes a boolean value and is per default turned off. If you want to turn this value on, you should do so since there is relatively few bad sides to not replying to broadcast pings.

3.4.3. icmp_ignore_bogus_error_responses

There are some routers on the internet and in other places that ignore the standards drawn up in RFC 1122 and sends out bogus responses to broadcast frames. Normally, these violations are logged via the kernel logging facilities, but if we do not want to see these error messages in our logs we may turn this variable on, which will lead to all such error messages being ignored totally. If you live close to such a router, you will save much harddrive space in not logging these messages.

This variable takes a boolean value as you may understand, and is per default set to 0 or off. If you need or want this option and want to get rid of some annoying error messages in your logs, you may turn this on.

3.4.4. icmp_ratelimit

icmp_ratelimit is the maximum rate at which the kernel generates icmp messages of the types specified by icmp_ratemask (see *icmp_ratemask*). The value is the number of jiffies the kernel has to wait between sending two such messages. Therefore zero means no limit. Typically 1 jiffy = 1/100 sec, so a value of 1 means no more than 100/sec, a value of 100 means no more than 1/sec.

Per default, this variable is set to 100, which means 1 ICMP packet may be sent in 100 jiffies. If we set it to 1, we allow 100 ICMP packets per second, and 0 means unlimited ICMP sendings.

3.4.5. icmp_ratemask

The icmp_ratemask variable sets the mask of which ICMP types should be ratelimited with the icmp_ratelimit variable. The types are put together in the mask by setting specific bits in this variable.

icmp_ratemask is the sum of 2^n for each icmp type you wish to ratelimit, and where n is each ICMP type as specified in the header files of the kernel. This ensures that each bit has a specific meaning. All of the different ICMP names and their corresponding values are available in the include file netinet/ip_icmp.h (generally `/usr/include/netinet/ip_icmp.h` on most systems). For more information about the different ICMP values and their meaning, see *RFC 792 - Internet Control Message Protocol*. This would be the complete mathematical formula for the expression:

$$\text{ratemask} = \text{SUM } 2^{n}$$

For all ICMP types n to be rate limited.

For example, if you want to include the ICMP Destination unreachable type in the ratemask, we would notice that this has the value 3 in the header files. To do the conversion then, we would calculate 2^3, which turns out to be 8. Finally, you would or this to the bitmask that you already have. So, if your mask already is 6160, you would go 6160+8 which should do the trick. To or two values means to only add the new entry, in case it was not added previously, in a binary way.

> **Caution**
>
> Make absolutely sure that the ratemask does not contain the bitmask that you want to add. If you fail to notice such a problem, your ratemask will become totally wrong. If you have 256 set already, and then try to set 256 again according to this description, you will wind up with ICMP Echo Request unset, and ICMP type 9 set. ICMP type 9 does not exist.

The default value of this variable is 6168, which means that ICMP Destination Unreachable, ICMP Source Quench, ICMP Time Exceeded and ICMP Parameter Problem is in the mask. ICMP Destination Unreachable equals 3, ICMP Source Quench equals 4, ICMP Time Exceeded equals 11 and ICMP Parameter Problem equals 12. Hence, the default value is calculated like this:

$$\text{ratemask} = 2^{3} + 2^{4} + 2^{11} + 2^{12}$$

Note: An attacker could cause a correctly operating host or router to flood a victim with ICMP replies by sending it packets that generate replies back to the (forged) source address of the victim. It is important in some cases to send such replies, but hardly ever important to generate them at a very high rate.

Tip: There is a small program to output the proper ICMP ratemask called ratemask, which is available at http://www.frozentux.net. It may be of some help when creating icmp_ratemasks, or if you want to find out what the current value means.

3.4.6. igmp_max_memberships

This variable changes the maximum amounts of multicast groups we may subscribe to per socket. This is per default set to 20 and may be changed as needed.

Fixme: NEED MORE DETAILS!!!

3.5. The conf/ variables

The conf directory is split up in several different directories which in turn contains a lot of settings that may be done. If you look closer at this directory listing you will see that most of the directories corresponds to your network interface names, such as eth0, tr0 or lo. These directories allows you to set different values depending on the different network interfaces. There are also the `all` and `default` directories which contains variables which will change the specific behaviour of all the different network interfaces. The listing of the conf directory may look like shown in the listing below.

```
root@firewall:/proc/sys/net/ipv4/conf# ls -l
total 0
dr-xr-xr-x    2 root     root            0 May  1 20:04 all/
dr-xr-xr-x    2 root     root            0 May  1 20:04 default/
dr-xr-xr-x    2 root     root            0 May  1 20:04 eth0/
dr-xr-xr-x    2 root     root            0 May  1 20:04 eth1/
dr-xr-xr-x    2 root     root            0 May  1 20:04 eth2/
dr-xr-xr-x    2 root     root            0 May  1 20:04 eth3/
dr-xr-xr-x    2 root     root            0 May  1 20:04 lo/
root@firewall:/proc/sys/net/ipv4/conf#
```

In the above configuration, we have 4 different ethernet adapters (eth0-3) and one lo adapter. Everyone should have the lo, or localhost, adapter since it is an integral part of most - if not all - unix systems, as long as it has any kind of network stack.

3.5.1. conf/DEV/, conf/all/ and conf/default/ differences

The `/conf/DEV/` directory, where DEV stands for some device or another, will only change the behaviour of the specific device in question. Now, `conf/all/` on the other hand will change the behaviour of *all* the other interfaces if changed.

The final directory named `conf/default/` will change the default values. This doesn't change the values in the already set up devices, but it will change the default values used for all the interfaces that may be brought up in the future. One usage would be if we set up a new interface eth0, change the `conf/eth0` variables for it, and finally set the defaults used. If we would then load five modems on ppp+, these variables would change since the default variables have changed.

3.5.2. accept_redirects

This variable tells your system whether it should accept ICMP redirects or not. ICMP redirects are normally used to tell a router, or sometimes hosts, that there is a better way to send the packets to specific hosts or networks, which is faster or is less congested.

This value takes a boolean value, and is turned off if it is set to 0 and turned on if it is set to 1. Per default, Linux does accept redirects, but I suggest you turn it off since it is generally considered as a security risk. Most machines should never have any specific requirements to accept being redirected, and hence you should mostly keep this setting off, unless you know that you will seriously need redirects once in a while.

3.5.3. accept_source_route

This variable tells the kernel if it should allow source routed packets or not. Source routed packets are generally looked upon as a security risk, and generally bad. For more information about source routing, see the *ip-param.txt* document.

This variable is per default turned on in all kernels. Of course, it takes a boolean value, and may be turned on (1) or off (0).

3.5.4. arp_filter

The arp_filter variable tells the kernel whether the IP address should be bound to a specific ARP address or not. The kernel decides to answer a specific packet incoming on a specific interface, if it decides that it would also send the reply back out through the same interface. This is what happens if you turn this option on. In general, it is a good idea to answer on an IP address bound to this computer be answered from whichever interface the packet was received on. However, in some cases, this may cause troubles.

In general, we should only turn this variable on if we are doing load-balancing, otherwise it should be turned off. Per default, this variable is turned off, since it is somewhat breaking the new thoughts about IP addresses. Previously, IP addresses where looked upon as a way of reaching a specific device on a piece of hardware, today, it is looked upon as a separate service in a way, which means that we should (and generally are) answering requests for a specific IP address, not depending on where we received it.

> **Tip:** For more information about ARP fluxing, I suggest that you read the information available in the *Guide to IP Layer Network Administration with Linux* document.

3.5.5. bootp_relay

The bootp_relay variable is supposed to accept packets with source address of 0.b.c.d destined not to this host as local packets. The BOOTP relay daemon will then catch these packets and forward them to the correct destination.

The bootp_relay variable takes a boolean value, and is per default turned off. It can either be turned on (1), or off (0). See the caution admonition before turning it on.

> **Caution**
>
> The bootp_relay variable is not implemented yet, hence this very short description. If you have needs of this, you should be welcome to implement it properly. If you want to implement this, you should get in touch with the netdev mailinglist to find out more specific information.

3.5.6. forwarding

This variable tells us if IP forwarding is turned on for a specific device. This could be used to turn on only forwarding between 2 devices, while the third is turned off. Hence, we limit which subnets has access to what. The value in this variable defaults to the same value as `ipv4/ip_forward`, so if you turn on ip_forward, all of these variables will change to 1 (on), and if you turn ip_forward off, all of the variables will be switched to 0 (off).

3.5.7. log_martians

This variable tells the kernel to log all packets that contains impossible addresses to the kernel logging facility. An impossible IP address may mean an IP address that we do not know how to contact, since the IP address is not contained in the routing tables.

> **Caution**
>
> The log_martians will increase verbosity under specific circumstances, but you should be aware that it is not as verbose as one may think. The main problems getting logged by this option are impossible redirects, bad classes, limited broadcasts or otherwise invalid according to the Forwarding Information Base (FIB). This may sound much, but the sircumstances are rather restrictive before anything gets logged.

Martian logging takes a boolean value, where 1 turns it on and 0 turns it off. Per default it is turned off.

3.5.8. mc_forwarding

This option turns on multicast routing for the specific device that we are configuring. To do this, the kernel must be configured and compiled with CONFIG_MROUTE. Additionally, you need a routing daemon that is available at AT&T Research FTP site. (ftp://ftp.research.att.com/dist/fenner/mrouted). This is a Multicast routing daemon that implements DVMRP. Another Multicast routing daemon that is available is the PIMd (http://netweb.usc.edu/pim/). PIMd implements the PIM (Protocol Independent Multicast) multicast routing protocol, in sparse mode. There are also links to other PIM-DM (Protocol Independent Multicast--Dense Mode) and PIM-SM (Protocol Independant Multicast--Sparse Mode) implementations from the last site. If you need more information about Multicasting under Linux, I suggest you read the Multicast HOWTO (http://www.tldp.org/HOWTO/Multicast-HOWTO.html) which contains tons of information about multicasting.

Briefly, The main usage of multicast is to send packets to several different hosts, on different networks. For example, a webpage is OK to send once to everyone who wants to see it, but if we want to have a video conference or video stream to several hundreds of users at once, we have two options of doing this. Either we send the packets once to every host, which means we will have to send hundreds of packets at the same time, and thus draining our bandwidth. The other solution is to use Multicast, in which case we only send one packet, and the packet will then be multiplexed along the road so that everyone who wants to receive the specific stream will get it.

This option takes a boolean value, and is per default turned off. Note that you do not need to turn it on, if you want your host to just listen to multicast packets. This is only needed if you want to forward multicast over the box.

3.5.9. proxy_arp

This variable sets Proxy ARP on or off in kernel for specific devices. Proxy ARP is a system of automatically answering ARP queries for other hosts, that may for example be located on other network segments that we have contact with. This may be necessary under certain circumstances, where other routers do not know how to reach specific networks or hosts. The Linux firewall/router may then answer the ARP queries on behalf of the hosts that we want to Proxy ARP for.

Proxy ARP is turned on for the network segment that we want to answer ARP queries for. We will then answer all ARP queries for that specific network or host, hence receiving the packets destined for the specific host, and we can then send them onwards to the real host.

The proxy_arp variables takes a boolean value. Per default, it is turned off, and may be turned on (1) or off (2) at will. If you want more information about Proxy ARP, read the Proxy-ARP mini HOWTO (http://www.ibiblio.org/pub/Linux/docs/HOWTO/unmaintained/mini/Proxy-ARP).

3.5.10. rp_filter

The rp_filter variable sets up a reverse patch (rp) filter on the specific interface. What this means, is quite simple. All it does, is to validate that the actual source address used by packets correlates properly with our routing table, and that packets with this specific source IP address are supposed to get their replies back through that interface again.

Caution

If you are using policy routing, or advanced routing, in one way or another, you are seriously suggested to turn the rp_filter variable off, since it may cause packets to be dropped. For example, if you have set up your routers to receive packets through one of them, and send outgoing packets through the other one. Now, if your webserver is connected through one interface to the incoming router, and one to the outgoing router, and the rp_filter variable is turned on, it will simply drop all incoming packets since the packets are not coming in to the webserver through the propriate interface in accordance to the routing table.

The variable takes a boolean value, and is per default turned off. However, a lot of Linux distributions turns on rp_filter through their startup scripts. Hence, if rp_filter is turned on, on your distribution and you want it turned off, start by looking at the `rc.d` scripts. The variable can either be turned off (0), or on (1).

Tip: The behaviour of the rp_filter variable is specified in *RFC 1812 - Requirements for IP Version 4 Routers* on pages 46-49 (section 4.2.2.11), page 55 (section 4.3.2.7) and page 90 (section 5.3.3.3). If you are doing serious routing, you should carefully read this document anyways.

3.5.11. secure_redirects

This variable turns on secure redirects. If it is turned off, the Linux kernel will accept ICMP redirects from any host, anywhere. However, if it is turned on, ICMP redirects will only be accepted from gateways listed in the default gateway list. This way we can get rid of most illegal redirects that can be used to log your traffic and grab sensitive data, such as passwords etcetera.

The secure_redirects variable takes a boolean value and is per default turned on. It may both be turned on or turned off. Note that this variable is overridden by the shared_media variable, so to turn this one on, you must turn on shared_media as well.

3.5.12. send_redirects

The send_redirects option tells the Linux kernel to send out ICMP redirects to other hosts. This should only be turned on, if the computer acts as a router of some sort. The ICMP redirects are mainly sent out to hosts, if we for example know that the other router/host should instead contact another server on their same subnet as the one we are receiving the packets on.

The send_redirects variable takes a boolean value and is per default turned on. It can take the values 0 (off) and 1 (on). In most cases where the computer is not running as a router of some kind, we could safely turn it off.

3.5.13. shared_media

The shared_media setting tells the kernel if the physical network connected to a specific network card is a shared media or not. For example, if several different IP networks with different netmasks operate over the same physical media or not. The main effect that this variable makes, is to tell the kernel whether it should send ICMP redirects to specific networks or not.

Per default this variable is turned on. It takes a boolean value, and may hence be turned on or off. Note that this variable overrides secure_redirects below.

3.6. Neigh reference

3.7. Netfilter reference

The variables available in /proc/sys/net/ipv4/netfilter/ all have to do with how netfilter and iptables behaves. As of writing this, these variables have not entered the mainstream kernel yet, but should within a couple of months. If you do not have access to these variables and want them, you need to patch your kernel with the tcp-window-tracking patch available in the iptables packet patch-o-matic. You can do this by running **make patch-o-matic KERNEL_DIR=/usr/src/linux** in the iptables package, and by making sure that you add that specific patch.

Previously, you had to change the connection tracking timings statically in the kernel source and then recompile. With this addition, it is much much simpler to change the default timeouts for different states among other things. You will also have the ability to make nefilter log out of window packets as well as packets with invalid window scale values. With this addition, netfilter and connection tracking is greatly enhanced and provides much better manageability.

In the rest of this section we will look closer at the different variables available in sysctl after adding this patch. All of the variables dealing with timeouts take their values in jiffies, or a 1/100th part of a second.

3.7.1. ip_ct_generic_timeout

This variable is used to tell the connection tracking code of netfilter the generic timeout in case we can not determine the protocol used and use more specific values. Any stream or packet that enters the firewall that can not be fully identified as any of the other protocol types in this section will get a generic timeout set to it.

The ip_ct_generic_timeout variable takes an integer value and is per default set to 600 seconds, or 10 minutes. If this is not enough for your applications, you should raise it until connections do not crash any longer or to an appropriate value.

3.7.2. ip_ct_icmp_timeout

The ip_ct_icmp_timeout variable is used to set the timeout for ICMP packets that will result in return traffic. This should in other words include Echo request and reply, Timestamp request and reply, Information request and reply and finally Address mask request and reply. Once we see one of the requests, we expect to see a return packet, and this is when the ICMP timeout comes into the picture.

As we would expect an ICMP reply to be quite quick, we expect the reply to have returned to, or through, the firewall within a couple of seconds. The ip_ct_icmp_timeout value is in other words rather low, approximately 30 seconds. This should generally be a good value, unless you live on an extremely bad connection.

3.7.3. ip_ct_tcp_be_liberal

This variable changes the behaviour in the state machine regarding TCP out of window packets. If this variable is turned off, all of the out of window packets are regarded as INVALID in the state machine. If it is turned on, the behaviour is much more liberal and only considers out of window RST packets as INVALID. It should generally be a good thing to go with this variable turned off, and should only be required to turn off during special occasions.

The ip_ct_tcp_be_liberal variable takes a boolean value and is per default set to 0, or turned off. All out of window packets are in other words considered as INVALID. As already stated, this should in most cases be the wanted behaviour.

3.7.4. ip_ct_tcp_log_invalid_scale

The ip_ct_tcp_log_invalid_scale variable is used to tell netfilter to log all invalid window scales used in a TCP packets. Window scaling is a specific TCP option used in most modern TCP implementations and is described in detail within the *RFC 1323 - TCP Extensions for High Performance*. If these scales are invalid for some reason, this variable tells netfilter to log the packet in question.

This variable takes a boolean value and is per default turned on, or set to 1 in other words. This should generally be a good thing, but if you notice that you get a lot of these log entries from some old host or other likely culprit you may as well turn this option off.

3.7.5. ip_ct_tcp_log_out_of_window

The ip_ct_tcp_log_out_of_window variable tells netfilter to log all packets that are considered as out of window. Generally, this is used in conjunction with the previously described ip_ct_tcp_be_liberal variable. All of the packets that are defined as out of window, depending on the setting of the ip_ct_tcp_be_liberal variable, will be logged if this variable is turned on.

The ip_ct_tcp_log_out_of_window variable takes a boolean value and is per default turned on, or set to 1. This should generally be a good idea, but if you do receive a lot of error messages due to errors that you can either not fix, or that is beyond your reach, you may as well turn this variable off.

3.7.6. ip_ct_tcp_timeout_close

This variable sets the timeout of the CLOSE state as defined in RFC 793. Briefly, the CLOSE state is entered if the client sends a FIN and then receive a FIN from the server, and replies with a FIN/ACK to the FIN sent from the server. The CLOSE state is then maintained until we receive a FIN/ACK replying the clients original FIN. When the final FIN/ACK arrives we enter the TIME-WAIT state.

The default timeout of this variable is set to 1000 jiffies, or 10 seconds. Generally, this should be a good value, but if you start noticing that a lot of clients gets hung in the CLOSE state, you should raise this value until the CLOSE state problem is no longer observed on the clients.

3.7.7. ip_ct_tcp_timeout_close_wait

This variable tells netfilter the timeout value of the CLOSE-WAIT state, which is also defined in the RFC 793. This state is entered if the client receives a FIN packet and then replies with a FIN/ACK packet. When this happens, the connection will enter a CLOSE-WAIT state. This state is then maintained until the client sends out the final FIN packet, at which time the connection will change state to LAST-ACK.

The default value of the ip_ct_tcp_timeout_close_wait variable is set to 43200 seconds, or 12 hours as of the tcp-window-tracking patch entering the kernel. Previously, this was per default set to 60 seconds, which in turn led to a lot of badly timed out connections. This timeout should in general be large enough to let most connections exit the CLOSE-WAIT state, but if you are having problems in the other direction that you are running out of conntrack entries, due to a lot of connections getting stuck in the CLOSE-WAIT state, your best bet is to either get more memory, or to bring this value down a few hours.

3.7.8. ip_ct_tcp_timeout_established

The ip_ct_tcp_timeout_established variable tells us the default timeout value for tracked connections in the ESTABLISHED state. All connections that has finished the initial 3-way handshake, and that has not seen any kind of FIN packets are considered as ESTABLISHED. This is in other words more or less the default state for a TCP connection.

Since we never want a connection to be lost on either side of the netfilter firewall, this timeout is set extremely high so we do not accidentally erase entries that are still used. Per default, the ip_ct_tcp_timeout_established variable is set to 432000 seconds, or 5 days.

Caution

You should not consider lowering this variable because you have too many active connection tracking entries going. If this is your problem, you should be looking at the ip_conntrack_core.c file, function ip_conntrack_init which defines the maximum connection tracking entries. Unfortunately, this maximum value is static as of writing this, and will probably remain so for a long time since each conntrack entry requires a non-swappable memory.

3.7.9. ip_ct_tcp_timeout_fin_wait

The ip_ct_tcp_timeout_fin_wait variable defines the timeout values for both the FIN-WAIT-1 and FIN-WAIT-2 states as described in RFC 793. The FIN-WAIT-1 state is entered when the server send a FIN packet, while the FIN-WAIT-2 state is entered when the server receives a FIN/ACK packet from the client in return to the initial FIN packet. If the server receive a FIN while still waiting for the FIN/ACK packet it enters the CLOSE (defined as CLOSING in RFC 793) state instead of FIN-WAIT-2.

The ip_ct_tcp_timeout_fin_wait variable is per default set to 120 seconds, or 2 minutes. The default here should generally be a good value but may be raised in case clients and servers are left in this state because the firewall stopped forwarding the packets due to the conntrack entries timing out in advance. Of course, if you have problems with conntrack running out of hash entries, this value may also be lowered, but not more than allowing the full closing handshake to take place on both ends.

3.7.10. ip_ct_tcp_timeout_last_ack

The ip_ct_tcp_timeout_last_ack variable sets the timeout value of the LAST-ACK state. This state is entered when the client sends a FIN packet in reply to an already received and replied FIN packet from the server. This state follows the CLOSE-WAIT state and is ended once we receive the final FIN/ACK packet in return to our own FIN packet. At this point, the conntrack entry is destroyed and we enter the CLOSED state (ie, no conntrack entry at all).

The default value of the ip_ct_tcp_timeout_last_ack variable is set to 30 seconds and should generally be more than sufficient for the last replying FIN/ACK packet, however, in certain cases this value may time out. If this is the case, you should try to raise this value. If you are having problems with running out of connection entries, you could possibly try to lower this value but avoid lowering it so much that your clients are left in a LAST-ACK state.

3.7.11. ip_ct_tcp_timeout_listen

This value is the initial state that all TCP receiving sockets are set to according to RFC 793. In other words, this state is reached for all sockets that sits and listens for connecting SYN packets. This state is never used in conntrack since conntrack can impossibly know about this state in advance since there is no network traffic indicating it. However, it is still made available to stay conformant with RFC 793 and perhaps for some other reasons that the netfilter core team may have.

This variable is per default set to 120 seconds, or 2 minutes. This could be lowered to 0 if you feel like it, but this *may* lead to strange behaviours and really bad problems. Since this value is never really used, you can as well leave it at the default value. It will not change any of the behaviours of netfilter as it looks now.

3.7.12. ip_ct_tcp_timeout_none

This variable is only used for an extremely brief period of time and indicates the connection before conntrack actually knows which state to put it in. For example, the firewall may never know for sure at what state a specific TCP connection is in when it sees the connection for the first time. When it sees the first packet, it sets the packet to state NONE. After this, it tries to decide at what state this connection is in according to RFC 793. Depending on what TCP flags the packet has set, netfilter can derive that the connection is in state SYN-SENT, ESTABLISHED, TIME-WAIT or CLOSE.

What all of the above really means, is that the connection is really never in the NONE state, or to be more accurate, it is only for an extremely brief timeperiod. There should in other words be no real need or usage in lowering this states timeout, since it is barely used anyways, and the packet should almost never fail to move on to another state from this state.

The ip_ct_tcp_timeout_none variable is per default set to 1800 seconds, or 30 minutes. This should generally pose no problem as already described and you have very little to win by changing this default value.

3.7.13. ip_ct_tcp_timeout_syn_recv

The ip_ct_tcp_timeout_syn_recv variable sets the timeout value for the SYN-RECEIVED (also known as SYN-RCVD or SYN-RECV) state as defined in RFC 793. The SYN-RECEIVED state is entered from the LISTEN or SYN-SENT state once the server receives a SYN packet and then replies with a SYN/ACK packet. The SYN-RECEIVED state is left for the ESTABLISHED state once the server receives the final ACK packet in the 3-way handshake.

The default value of the ip_ct_tcp_timeout_syn_recv variable is 60 seconds, or 1 minute. This should generally be a good value, but if you do experience timeouts where your server or clients end up in a SYN-RECV or SYN-SENT state, you should consider raising this value a bit. It is generally a bad idea to lower this variable to get over any problems with connections timing out.

3.7.14. ip_ct_tcp_timeout_syn_sent

The ip_ct_tcp_timeout_syn_sent state sets the timeout of the SYN-SENT state as defined in RFC 793. The SYN-SENT state is entered once the client has sent out a SYN packet and is still awaiting the returning SYN/ACK packet. The SYN-SENT state is then left for the SYN-RCVD or ESTABLISHED states once we get the SYN/ACK packet.

The default value of the ip_ct_tcp_timeout_syn_sent variable is set to 120 seconds, or 2 minutes. This should generally be a good setting, but if you experience problems where the client and server gets stuck in a SYN-SENT or SYN-RCVD state, you may possibly need to raise this value. It is generally a bad idea to lower this value to get around problems with not having enough conntrack entries.

3.7.15. ip_ct_tcp_timeout_time_wait

The ip_ct_tcp_timeout_time_wait variable defines the timeout value of the TIME-WAIT state as defined by RFC 793. This is the final state possible in a TCP connection. When a connection is closed in both directions, the server and client enters the TIME-WAIT state, which is used so that all stale packets have time to enter the client or server. One example of the usage of this may be if packets are reordered during transit between the hosts and winds up in a different order at either side. In such a case, the timewindow defined in the ip_ct_tcp_timeout_time_wait variable is used so that those packets may reach their destinations anyways. When the timeout expires, the conntrack entry is destroyed and we enter the state CLOSED, which means that there is no conntrack data at all for the connection in question.

The default value of the ip_ct_tcp_timeout_time_wait variable is set to 120 seconds, or 2 minutes. You should generally want to keep this value if you know that you live on a connection that is slow and that often reorders the packets in question. If this value is too low you will often experience corrupt downloads or missing data in downloaded data, you should definitely avoid tuning this value down in such a case, and you may most definitely consider raising the timeout of this value in such case. If you never experience such behaviour or any other problems like that, you may most probably lower this value so that conntrack entries die faster, and hence recycle the conntrack entry space faster.

3.7.16. ip_ct_udp_timeout

The ip_ct_udp_timeout variable specifies the timeout for initial UDP packets in a connection. When a UDP connection is initialized, the UDP packet enters an NEW and then ESTABLISHED state once it has seen return traffic to the original UDP packet. However, it maintains the same timeout until it has seen several packets go back and forth and becomes assured, at which point it is instead considered a stream.

While this initial state is maintained, the default timeout is 30 seconds. If you are using UDP protocols that send very little data during longer timeframes, you should consider raising this value so that the state machine is able to keep track of your connections properly. It is generally a bad idea to lower this, unless you know that your hosts sends UDP packets very often and don’t expect a lot of late replies, which would mean a lot of unnecessary open conntrack entries.

3.7.17. ip_ct_udp_timeout_stream

The ip_ct_udp_timeout_stream variable specifies the timeout values of the UDP streams once they have sent enough packets to reach the assured state. This state is normally reached for connections that send a lot of data and relatively often, such as streaming services or ICQ. Examples of streaming services may be certain realplayer servers, or speak freely. This value should always be larger than the initial timeout value for UDP streams since it is used on connections that we know for sure expects a lot of traffic back and forth, even though it may not be very often.

The ip_ct_udp_timeout_stream variable is per default set to 180 seconds, or 3 minutes in recent kernels.

If you are having problems with connections timing out, you may try raising this value a bit. It is generally a bad idea to lower this value, since the connection will be destroyed once it times out from this state. Unfortunately, UDP is a stateless protocol, so it is very hard to derive any specific states of the connections. Because of this, there is no specific conntrack timeouts for UDP streams that are about to close, or that has closed.

3.8. Route reference

The route/ section of the IPv4 sysctl's contains mainly some loosely related variables that changes the effect in the route cache, or the routing code in the kernel. Some will simply change how many error/warning messages are printed by the routing code, and others will change the garbage collection timings in the route cache.

3.8.1. error_burst

This variables is used in conjunction with **error_cost** to limit how many ICMP destination unreachable messages the routing code will send out. This variable tells the function the maximum amount of tokens that we have available, while the **error_cost** variable tells us how many tokens are used up by each ICMP destination unreachable packet. If **error_burst** is emptied and we receive more packets for that host, we will ignore them.

> **Note:** ICMP destination unreachable messages are generally sent when our host, or router, does not know how to reach a specific host or network. This may be for three basic reasons:
>
> 1. We can not reach the nexthop to the destination.
> 2. We have no route set up for the network segment or host.
> 3. We have a routing rule or route set to unreachable, throw or prohibit.
>
> During these three sircumstances, the networking code sends out ICMP Destination Unreachable messages with three different codes if necessary:
>
> 1. ICMP host unreachable in case the host should be directly connected to a network we are part of, or if the host was set to unreachable or throw through a rule or route.
> 2. ICMP network unreachable if we do not know how to reach the network in question, or if the network was set to unreachable or throw through a rule or route.
> 3. ICMP communication administratively prohibited by filtering is sent if we have a rule or route set to prohibit.

The **error_burst** variable is per default set to 500 and takes an integer value. Together with the default value of **error_cost** this means that we allow the routing code to send 5 ICMP Destination unreachables per second.

3.8.2. error_cost

For a complete explanation, see the *error_burst* variable. Basically, this variable sets the cost of sending a single ICMP destination unreachable to someone.

The **error_cost** variable is per default set to 100, which means we can send a maximum of 5 ICMP destination unreachables per second, together with the default value of **error_burst**.

3.8.3. flush

This variable is extremely simple in use. It can not be read since it contains no real data. If you write anything to it as root, it will simply flush the whole routing cache. For example, if you know that a route has changed somewhere, you would simply echo something into this file, and the route cache will be emptied. If you would like to read the route cache, take a look at the `/proc/net/rt_cache`. This file contains a lot of information about the routes currently in the cache.

3.8.4. gc_elasticity

3.8.5. gc_interval

3.8.6. gc_min_interval

3.8.7. gc_thresh

3.8.8. gc_timeout

3.8.9. max_delay

3.8.10. max_size

3.8.11. min_adv_mss

3.8.12. min_delay

3.8.13. min_pmtu

3.8.14. mtu_expires

3.8.15. redirect_load

3.8.16. redirect_number

3.8.17. redirect_silence

Appendix A. Measurements used in kernel

A.1. Jiffies

Jiffies are the internal timeunits of the Linux kernel. This is based on the HZ definition that you can find in /usr/include/asm/param.h on your own system, and differs from different hardware architectures. For example, on the i386 system this is set to 1/100s, and on the Alpha hardware it is set to 1/1024s. For a complete listing of the different values on different hardware architectures, see below.

Table A-1. Jiffies on different hardware

Architecture	Jiffies	Notes
Alpha	1024	It seems to me that only AlphaServer 1200, 4000 and 4100 are set to 1024, if CONFIG_ALPHA_RAWHIDE is configured. If CONFIG_ALPHA_RAWHIDE is not set, then HZ is set to 1200. If anyone knows for sure about this, they are welcome to contact me.
ARM	100	
CRIS	100	
i386	100	
ia64	1024	Unless CONFIG_IA64_HP_SIM is configured. Then HZ is defined to 32 since we will then simulate ia64 architecture, which is slow.
m68k	100	
MIPS	100	You should better read the source of include/asm/param.h, since this value vary quite a lot based on what kind of hardware you are running.
MIPS64	100	
PA-RISC	100	
PPC	100	
PPC-64	100	Once again, if you run this hardware, read the include/asm/param.h to fully understand this value.
S390	100	
S390X	100	
SH	100	
Sparc	100	
Sparc64	100	

Appendix B. Other resources

This is a list of different sources of information that I have used to get more information about different behaviours and so on. Some resources may have been left out on purpose while others where left out on mistake. If you have any well written resource, let me know and I may add it here.

- RFC 791 - Internet Protocol (http://ipsysctl-tutorial.frozentux.net/other/rfc791.txt) - The original document specifying the IP protocol. This document was edited by J. Postel from the DoD Standard Internet Protocol.
- RFC 792 - Internet Control Message Protocol (http://ipsysctl-tutorial.frozentux.net/other/rfc792.txt) - The definitive resource for all information about ICMP packets. Whatever technical information you need about the ICMP protocol, this is where you should turn first. Written by J. Postel.
- RFC 793 - Transmission Control Protocol (http://ipsysctl-tutorial.frozentux.net/other/rfc793.txt) - This is the original resource on how TCP should behave on all hosts. This document has been the standard on how TCP should work since 1981 and forward. Extremely technical, but a must read for anyone who wants to learn TCP in every detail. This was originally a Department of Defense standard written by J. Postel.
- RFC 959 - File Transfer Protocol (http://ipsysctl-tutorial.frozentux.net/other/rfc959.txt) - This is the RFC documenting the FTP protocol and how it works. Originally written by J. Postel and J. Reynolds.
- RFC 1072 - TCP Extensions for Long-Delay Paths (http://ipsysctl-tutorial.frozentux.net/other/rfc1072.txt) - This RFC discusses possible solutions to specific problems in paths or streams that have extremely long round-trip delay or extremely high bandwidth and what can be done about TCP sequence number reusing among other things. This RFC was obsoleted by RFC 1323, but is available here since it contains valuable information. This RFC was written by V. Jacobson and R. Braden.
- RFC 1122 - Requirements for Internet Hosts -- Communication Layers (http://ipsysctl-tutorial.frozentux.net/other/rfc1122.txt) - This RFC specifies requirements for hosts that are working on the Internet. Edited by R. Braden and written by IETF.
- RFC 1123 - Requirements for Internet Hosts -- Application and Support (http://ipsysctl-tutorial.frozentux.net/other/rfc1123.txt) - This document describes the requirements for all internet hosts regarding the Application layer protocol, and all of their supporting protocols and layers. It was edited by R. Braden
- RFC 1323 - TCP Extensions for High Performance (http://ipsysctl-tutorial.frozentux.net/other/rfc1323.txt) - This RFC obsoletes RFC 1072 previously described in this section. This RFC gives a brief explanation of possible problems with connections that have extremely long round-trip delays and/or extremely high bandwidth, as well as a set of solutions to this problem. This RFC was written by V. Jacobson, R. Braden and D. Borman.
- RFC 1337 - TIME-WAIT Assassination Hazards in TCP (http://ipsysctl-tutorial.frozentux.net/other/rfc1337.txt) - This RFC discusses a few possible failures that TCP connections may get into. It also discusses a few remedies to these problems, and identifies a simple fix that can be used. This RFC was written by R. Braden.

- RFC 1812 - Requirements for IP Version 4 Routers (http://ipsysctl-tutorial.frozentux.net/other/rfc1812.txt) - RFC 1812 discusses the requirements of all IPv4 routers. This is a must read if you are in any way serious about your routing, or if you need to implement anything that has to do with routing. This RFC is based on 1716, which it also obsoletes, and was updated by F. Baker.
- RFC 2018 - TCP Selective Acknowledgement Options (http://ipsysctl-tutorial.frozentux.net/other/rfc2018.txt) - This RFC discusses the insertion of a new TCP Option called SACK, or Selective Acknowledgement, which gives better possibilities to acknowledge parts of a TCP window. Written by M. Mathis, J. Mahdavi, S. Floyd and A. Romanow.
- RFC 2883 - An Extension to Selective Acknowledgement (SACK) Option for TCP (http://ipsysctl-tutorial.frozentux.net/other/rfc2883.txt) - This RFC extends RFC 2018 and defines how to acknowledge duplicate incoming packets and how to handle these duplicate packets. Written by S. Floyd, J. Mahdavi, M. Mathis and M. Podolsky.
- RFC 2884 - Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks (http://ipsysctl-tutorial.frozentux.net/other/rfc2884.txt) - This is a performance evaluation of ECN and how it works in IP networks. This is an informational RFC only, and contains no specific information on how ECN works. The RFC was written by J. Hadi Salim and U. Ahmed.
- RFC 3168 - The Addition of Explicit Congestion Notification (ECN) to IP (http://ipsysctl-tutorial.frozentux.net/other/rfc3168.txt) - This RFC defines how ECN is to be used on a technical level and how it should be implemented in the TCP and IP protocols. Written by K. Ramakrishnan, S. Floyd and D. Black.
- Guide to IP Layer Network Administration with Linux (http://linux-ip.net/) - A large guide covering how IP works in Linux, and how to administrate it. It covers most tools, and common IP concepts that may arise. Especially look closer at the ARP-flux problems covered, which is a very good explanation.
- ECN-under-Linux Unofficial Vendor Support Page (http://gtf.org/garzik/ecn/) - A great page which contains information about the ECN problems, and a brief list of the most common hardware that is problematic.
- ECN: Executive Summary (http://www.uwsg.indiana.edu/hypermail/linux/kernel/0009.1/0329.html) - A mail sent to the Linux kernel mailing list by Dax Kelson claiming 8% of the Internet is unreachable with ECN enabled. This mail is dated in 2000, so this has changed much by today.
- TCP Tuning Guide (http://www-didc.lbl.gov/TCP-tuning/TCP-tuning.html) - A document discussing different measures one may take to tune TCP networks for better performance. It was written by a project group lead by Brian L. Tierney, DIDC (Data Intensive Distributed Computing group) at Lawrence Berkeley Labs. Most of this information does not apply to networks slower than giga ethernet.
- services.txt (http://ipsysctl-tutorial.frozentux.net/other/services.txt) - This is an example `services` file, which can be found in `/etc/services` on most Unix systems. This `services` file was grabbed from a Slackware 8.0 system.
- protocols.txt (http://ipsysctl-tutorial.frozentux.net/other/protocols.txt) - This simple file is normally hosted in every unix machine as `/etc/protocols`. The `protocols` file contains name and numbers representing all the different protocols in the IP layer. This file was taken from a default Slackware 8.0 machine.
- ip-sysctl.txt (http://ipsysctl-tutorial.frozentux.net/other/ip-sysctl.txt) - This is the classical documentation for the ip-sysctl functions in the linux kernel. This document was taken out of the linux

kernel 2.4.14 and has not evolved a lot in the last couple of months, as of writing this.

- ip_dynaddr.txt (http://ipsysctl-tutorial.frozentux.net/other/ip_dynaddr.txt) - This file contains information about the IPv4 ip_dynaddr option available in sysctl. This file was distributed with the linux kernel 2.4.14 and was originally written by JuanJo Ciarlante.
- ip-param.txt (http://ipsysctl-tutorial.frozentux.net/other/ip-param.txt) - This document contains information about IP Options mainly, and which RFC's to look in, to get more information about the specific IP options.
- netdev mailinglist (http://oss.sgi.com) - The netdev mailinglist is the main mailing list for the network developers in Linux. This is the final stop if you find something you don't understand or don't know how it works.

Appendix C. History

Version 1.0.4 (21 May 2003)
http://ipsysctl-tutorial.frozentux.net
By: Oskar Andreasson

Version 1.0.3 (26 Apr 2003)
http://ipsysctl-tutorial.frozentux.net
By: Oskar Andreasson
Contributors: Don Cohen, Martin A. Brown and Brian Tierney.

Version 1.0.2 (19 Dec 2002)
http://ipsysctl-tutorial.frozentux.net
By: Oskar Andreasson
Contributors: Alan Cox, Michael T. Babcock, Don Cohen and several others.

Version 1.0.1 (23 Oct 2002)
http://ipsysctl-tutorial.frozentux.net
By: Oskar Andreasson
Contributors: William Stearns, Don Gould, Mattias Webjörn Eriksson and Michael Bussmann.

Version 1.0.0 (8 Oct 2002)
http://ipsysctl-tutorial.frozentux.net
By: Oskar Andreasson
Contributors: Jozsef Kadlecsik, Nivedita Singhvi, JuanJo Ciarlanti and Alexey Kuznetsov

Appendix D. GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject.

(For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Appendix E. GNU General Public License

Version 2, June 1991

0. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

1. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

1. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

2. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

3. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
   1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
   2. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
   3. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty

(or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

4. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   A. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   B. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   C. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

5. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
6. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
7. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
8. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.
11. NO WARRANTY

    BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

2. How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program

'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.