NAME

tcpstat − report network interface statistics

SYNOPSIS

tcpstat [**−?haeFlp**] [**−B** *bps*] [**−b** *bps*] [**−f** *filter expr*] [**−i** *interface*]
[**−o** *output*] [**−R** *seconds*] [**−r** *filename*] [**−s** *seconds*] [*interval*]

DESCRIPTION

tcpstat reports certain network interface statistics much like vmstat(8) does for system statistics. Statistics include bandwidth being used, number of packets, average packet size, and much more.

Network information is collected either by reading data from *filename*, or by directly monitoring the network interface *interface*. The default action for **tcpstat** is to automatically search for an appropriate interface, and to show current statistics on it.

interval is the sample interval, in seconds, in which the statistics are based upon and when in default mode, how often the display is updated. If -1 is given, then the interval is taken to be the entire length of the sample. Default is 5 seconds.

When reading data from *filename*, **tcpstat** will exit immediately after the entire file has been processed. When collecting data from *interface*, **tcpstat** will keep running unless the **−s** option had been specified.

OPTIONS

The options are as follows:

−a Accounting mode. Displays the estimated number of bytes per second, minute, hour, day, and month.

−b *bps* Bandwidth mode. Displays the total number of seconds the data-throughput exceeded *bps*, and the percentage of total time this was, as if the interface were limited to *bps* bits per second. See the **NOTES** section below to see how the *interval* affects bandwidth calculation.

−B *bps* "Dumb" bandwidth mode. Displays the total number of seconds the data-throughput exceeded *bps*, and the percentage of total time this was. See the **NOTES** section below to see difference between "dumb" and normal bandwidth modes.

−e Suppresses the display of empty intervals.

−F Flush all output streams after printing each interval. Sometimes useful when redirecting output into a file, or piping tcpstat into another program like grep(1).

−f *filter expr*
Filter the packets according the rules given by *filter expr*. For the syntax of these rules, see tcpdump(1). The argument must be quoted if it contains spaces in order to separate it from other options.

−h, **−**? Display version and a brief help message.

−i *interface*
Do a live capture (rather than read from a file) on the interface *interface* given on the command line. If *interface* is "auto" then **tcpstat** tries to find an appropriate one by itself.

−l Include the size of the link-layer header when calculating statistics. (Ethernet only, right now. Usually 14 bytes per packet.)

-p Set the interface into non-promiscuous mode (promiscuous is the default) when doing live captures.

-o *format*
Set the output format when displaying statistics. See the **OUTPUT FORMAT** section below for a description of the syntax.

-R *seconds*
Show the timestamp relative to *seconds*. Avoid this option, because it will most likely go away in future versions.

-r *filename*
Read all data from *filename*, which may be a regular file, a named pipe or "-" to read it's data from standard input. Acceptable file formats include pcap (tcpdump(1) files) and "snoop" format files. *filename* is usually a file created by the tcpdump(1) command using the "-w" option.

-s *seconds*
When monitoring an interface, **tcpstat** runs for only *seconds* seconds, and then quits. When reading from a data file, **tcpstat** prints statistics for *seconds* seconds relative to the first packet seen.

OUTPUT FORMAT

The *output* string is any quoted string, and **tcpstat** will write this string to the stdout. In addition, **tcpstat** will substitute certain values for substrings which begin with a "%", as well as most standard printf(3) "\" escape characters. Here is a list of all substitution strings:

%A	the number of ARP packets
%a	the average packet size in bytes
%B	the number of bytes per second
%b	the number of bits per second
%C	the number of ICMP and ICMPv6 packets
%d	the standard deviation of the size of each packet in bytes
%I	the number of IPv4 packets
%l	the network "load" over the last minute, similar to uptime(1)
%M	the maximum packet size in bytes
%m	the minimum packet size in bytes
%N	the number of bytes
%n	the number of packets
%p	the number of packets per second
%R	same as %S, but relative to the first packet seen
%r	same as %s, but relative to the first packet seen
%S	the timestamp for the interval in seconds after the "UNIX epoch"
%s	the timestamp for the interval in seconds.microseconds after the "UNIX epoch"

%T the number of TCP packets

%U the number of UDP packets

%V the number of IPv6 packets

%*number*
switch the output to the file descriptor *number* at this point in the string. All output for each interval before this parameter is by default the standard output (file descriptor 1). Useful when redirecting the output into more than one file (or fifo) for separate statistics. Be sure you know where they are going. Writing to "dangling" file descriptors (without directing them to a specific destination) may produce unexpected results.

%% the "%" character

The default *format* string for **tcpstat** is:

"Time:%S\tn=%n\tavg=%a\tstddev=%d\tbps=%b\n"

which will produce an output which would look similar to:

```
Time:940948785 n=107   avg=251.81     stddev=422.45  bps=43110.40
Time:940948790 n=99    avg=400.21     stddev=539.39  bps=63393.60
Time:940948795 n=43    avg=257.16     stddev=352.83  bps=17692.80
```

It is worth noting for example, that many of the protocol filters (%T, %U, etc.) may be seen as being redundant because protocols can be filtered using **-f** (see **OPTIONS** above)

SIGNALS

Upon receiving a SIGINT, **tcpstat** will print any remaining statistics, and then exit. Upon receiving a SIGUSR1 when printing intervals, **tcpstat** will print the current statistics immediately. This can be useful when using an interval length of "-1" to print statistics on demand.

FILES

`/dev/bpf`**n** the packet filter device

EXAMPLES

```
tcpstat -i fxp0
```

Displays the default statistics every 5 seconds of all traffic currently passing through the fxp0 network interface.

```
tcpstat -r file.dump
```

Displays the default statistics every 5 seconds from the `tcpdump`(1) generated file "file.dump".

```
tcpstat -f 'port (smtp or http)' -o '%S %b\n' -r file.dump 2.3
```

Displays every 2.3 seconds the timestamp together with smtp and http traffic throughput of the data from "file.dump", in a format which would be suitable for `gnuplot`(1).

```
tcpstat -b 28800 -r file.dump 0.5
```

Displays what percentage of the traffic in file.dump exceeded the speed of my modem (28800 bits per second.)

SEE ALSO

`tcpdump`(1), `pcap`(3), `bpf`(4), `printf`(3)

NOTES

Interval size affects bandwidth

Due to the nature of how bandwidth is actually measured (from discrete samples of data), the bandwidth numbers displayed will vary according to the *interval* variable. Generally speaking, if you often have rapid bursts of packet data, the bandwidth reported will not reflect this when *interval* is sufficiently large. This results in an "averaging" effect, which may or may not be desired. On the other hand, if *interval* is too small (say < 0.01), this results in unrealisticaly large bandwidths for very short amounts of time.

The reason for the latter is that most network interfaces do not hand over packets bit by bit, but rather packet by packet. Thus, each packet is reported as being tranfered "instantaneously", resulting in "infinite" (or rather indeterminable) bandwidth. Thus, when counting single bits on the wire, there really is no such thing as "bandwidth" because they aren't really moving from the network stack's point of view (cf. Zeno's Paradox.)

A possible solution is to internaly spline the packet sizes together and report the bandwidth as the scalar integral over the given interval, but this has yet to be implimented, and to be honest, would be the proverbial cruise missle to destroy an ant hill.

That being said (whew!), a "good value" for *interval* is usualy somewhere between 0.5 and 2.

Difference between normal and 'dumb' bandwidth modes.

In normal bandwidth mode, when an interval exceeds the given bandwidth, the extra bytes are "moved" into the next interval. This has the effect of trying to imagine how overloaded an interface would be if the interface had a smaller bandwidth, yet same amount of data tried to get through.

In "dumb" bandwidth mode, each interval which exceeds the given bandwidth is simply counted. Nothin' else.

HISTORY

tcpstat was first written in Winter 1998 using FreeBSD 3.0, and then finaly retrofited for Linux in Spring 2000.

AUTHORS

Paul Herman ⟨pherman@frenchfries.net⟩
Cologne, Germany.

Please send all bug reports to this address.

BUGS

Due to a bug in libpcap, tcpstat will hang indefinately under Linux when no packets arrive. This is because the timeout in pcap_open_live() is ignored under Linux when the interface is idle, which causes pcap_dispatch() to never return.

Not tested with link types other than Ethernet, PPP, and "None" types.

There may be problems reading non-IPv4 packets across platforms when reading null type link layers. This is due to a lack of a standardized packet type descriptor in libpcap for this link type.

Snoop file formats cannot be read from stdin or named pipes.