NAME

gulp − drink efficiently from the network firehose

SYNOPSIS

gulp [**--help** | *OPTIONS*]

DESCRIPTION

On a system with at least two CPUs (or cores), **Gulp** will probably drop far fewer packets than **tcpdump** when capturing from ethernet and writing to disk, allowing for much higher packet capture rates. **Gulp** has the ability to read directly from the network but even piping output from legacy applications through **gulp** before writing to disk will probably result in a substantial performance improvement.

Since **Gulp** uses CPUs #0-1, if you use **Gulp** in a pipeline and have more than 2 CPUs, you can further improve performance by explicitly running other programs on CPUs #2-N with **taskset**(1) as shown in some examples below.

To improve interactive response at low packet rates, **Gulp** will flush its ring buffer if it has not written anything in the last second. If the data rate increases, Gulp will realign its writes to even block boundaries for optimum writing efficiency.

When **Gulp** receives an interrupt, it will stop filling its ring buffer but will not exit until it has finished writing whatever remains in the ring buffer. If the buffer is large this can take a while--be patient.

OPTIONS

−d Decapsulates packets from a Cisco "Encapsulated Remote SPAN Port" (ERSPAN). Sets the pcap filter expression to "proto gre" and strips off Cisco GRE headers (50 bytes) from the packets captured. (If used with "-f" note that arguments are processed left to right).

−f Specify a pcap filter expression. This may be useful to select one from many GRE streams (if using **−d**), or if not using **−d**, because filtering out packets in the kernel is more efficient than passing them first through **Gulp** and then filtering them out.

−i *eth#* Specify the network interface to read from. The default is **eth1** or the value of environment variable $CAP_IFACE, if present. Specifying "−" as an "interface" reads a pcap file from standard input instead. (If you forget -d during a live capture, you can decapsulate offline this way).

−r *#* Specify a ring buffer size (in megabytes). Values from 1-1024 are permitted, the default is 100MB. If possible, the ring buffer will be locked into RAM.

−c Just copy and buffer bytes from stdin to stdout -- don't read packets from the network and don't assume anything about the format of the data. This may be useful to improve the real-time performance of another application.

−s *#* Packet capture snapshot length. By default, complete packets are captured. For efficiency, captured packets can be truncated to a given length during the capture process, which reduces capture overhead and pcap file sizes. (If used with "-d", specifies length after decapsulation.)

−F Skip default check that the capture interface is an Ethernet interface.

−x Use file locking to request (via exclusive lock) that this be the only instance of **Gulp** running. If other instances are already running, they must be stopped before Gulp will start with this option.

−X Override an exclusive lock (above) and run anyway. An instance of **Gulp** started this way will hold a shared lock if no exclusive locks were broken, otherwise it will hold no locks at all (causing a subsequent attempt to get an exclusive lock to succeed).

−v Print program version and exit.

−V xxxxxxxxxx

If the string of Xs is wide enough (10 or more), it will be overwritten twice per second with a brief capture status update consisting of one digit followed by two percentages. The digit is the number of decimal digits in the actual count of lost packets (0 indicates no drops). The two percentages

are the current and maximum ring buffer utilization. The updated argument string can be seen with the "**ps −x**" command (or equivalent).

If the string of Xs is too short to hold the information above, a more verbose status line is written to standard error instead (also twice/second). The first method is probably more useful to occasionally check on long captures and the second will be more convenient while experimenting and setting up a capture.

−p *#* Specify the thread polling interval (in microseconds). The reader/writer threads poll at this interval when the ring buffer is full/empty waiting for that to change. Polling (even frequently) on modern hardware consumes immeasurably few resources. The default interval is 1000 (microseconds).

−q Suppress warnings about the ring buffer being full. (If input is not from a live capture, no data will be lost when the ring buffer fills so the warning can be safely suppressed. (If stdin is actually a file, warning suppression will happen automatically.)

−z *#* Specify output write blocksize. Any power of two between 4096 and 65536 will probably be OK. It seems to be slightly more efficient to write larger blocks so the default is 65536 for now.

CAPTURE TO FILE OPTIONS

−o *dir* Redirects pcap output into a collection of files in directory *dir*. Pcap files will be named **pcap***###* (where *###* starts at 000 and counts up). To prevent mischief, the directory must exist (and be writable by the user running Gulp if Gulp is running setuid).

−n *name*
Specify a filename prefix of *name* instead of the default **pcap**.

−t When set, the current timestamp will be appended after *name*. This option implicitely sets **−z 1000** (and so always produces filenames like *name*1234567890.*###*).

−C *#* When using the -o option above, start a new pcap file when the old one reaches about # times the size of the ring buffer. The default value is 10 and the default ring buffer size is 100MB so by default, pcap files will grow to about 1000MB before a new one is started. Since some programs read an entire pcap file into memory when using it, splitting the output into chunks can be helpful.

−G *#* If set, specifies the maximum delay (in seconds) after which a new output file is created (same behavior as the equivalent **tcpdump** option).

−W *#* Specifies a maximum number of pcap files to create before overwriting them. The default is to never overwrite them. This option allows capturing indefinitely (waiting for a problem to occur) with finite disk space.

−Z *postrotate-command*
If set, specifies a command to post-process each capture file after they got rotated.

OTHER OPTIONS

−B This option is of academic interest only. It enables code to check before each write whether select(2) thinks the write would block. When **Gulp** exits, it announces whether any writes would have blocked. On linux, no matter how long writes to disk actually take, select(2) never says they will block. If you pipe the output of **Gulp** through cat before writing to disk, select *will* detect writes to the pipe would block.

−Y This option is of academic interest only. Writes which would block are deferred until select(2) says they won't block.

EXAMPLES

In the examples below, the ellipsis (...) refers to Berkeley Packet Filter (pcap) expressions, such as "**host foo".**

1) reduce packet loss of a tcpdump packet capture:

(gulp -c works in any pipeline as it does no data interpretation)

tcpdump -i *eth1* **-w - ... | gulp -c >** *pcapfile*
or if you have more than 2, run tcpdump and gulp on different CPUs
taskset -c 2 tcpdump -i *eth1* **-w - ... | gulp -c >** *pcapfile*

(gulp uses CPUs #0,1 so use #2 for tcpdump to reduce interference)

2) same as above but more efficiently using gulp itself to capture:

gulp -i *eth1* **-f "..." >** *pcapfile*

3) capture and decapsulate an ERSPAN feed and save the result to disk:

gulp -i *eth1* **-d >** *pcapfile*

4) capture, decapsulate and then filter with tcpdump before saving:

gulp -i *eth1* **-d | tcpdump -r - -s0 -w** *pcapfile ...*
or if you have more than 2 CPUs
gulp -i *eth1* **-d | taskset -c 2 tcpdump -r - -s0 -w** *pcapfile ...*

5) capture everything to disk; then decapsulate offline:

gulp -i *eth1* **>** *pcapfile1***; gulp -d -i - <** *pcapfile1* **>** *pcapfile2*

6) capture, decapsulate and then filter with ngrep:

gulp -i *eth1* **-d | ngrep -I - -O** *pcapfile regex ...*

7) capture, decapsulate and feed into ntop:

gulp -i *eth1* **-d | ntop -f /dev/stdin -m** *a.b.c.d/x* **...**
or if using ntop's -u flag:
mkfifo *pipe***; chmod 644** *pipe* # the first time only
gulp -i *eth1* **-d >** *pipe* **& ntop -u** *ntop* **-f** *pipe* **-m** *a.b.c.d/x* **...**

8) capture, decapsulate and feed into WireShark:

gulp -i *eth1* **-d | /usr/sbin/wireshark -i - -k**

9) capture to 1000MB files, keeping just the most recent 10 (files):

gulp -i *eth1* **-C 10 -W 10 -o** *pcapdir*
or with help from tcpdump:
gulp -i *eth1* **| taskset -c 2 tcpdump -r- -C 1000 -W 10 -w** *pcapname*

BUGS

On some systems, one interrupt may not break out of the pcap library's inner packet capture loop (if no packets arrive matching the filter expression). In that case, a second interrupt should do the trick.

On a busy network, Gulp may drop a few packets at startup while it is initializing. This makes Gulp look bad but is probably not a problem in practice.

AUTHOR

Written by Corey Satten, corey @ u.washington.edu

See **http://corey.elsewhere.org/gulp/** for more information and the latest version.

This manpage corresponds to Gulp version 1.58-crox.

COPYRIGHT

LICENSE

SEE ALSO

tcpdump(8), wireshark(1), ngrep(8), tcptrace(1), tcpflow(1), ntop(8) taskset(1) and **pcap(3).**