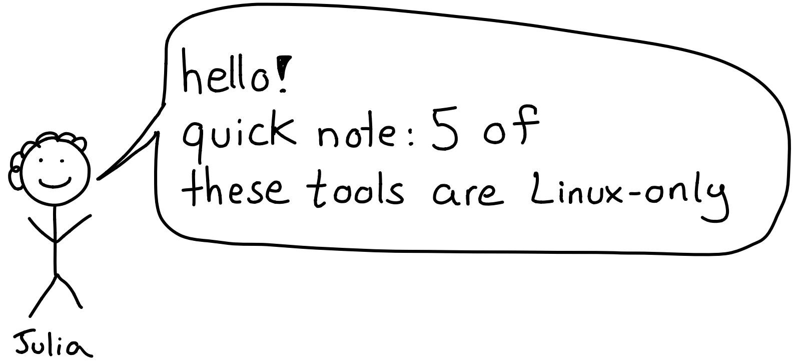
Bite Size Networking

by Julia Evans



Linux only tool	BSD/Mac equivalent
ip	ifconfig, route
tc	dummynet (?) (BSD)
ss	netstat
iptables	pf (BSD)
ethtool	ifconfig, kind of (?)

♥ Table of contents ♥

dig

dig makes DNS queries!

```
$ dig google.com
```

answers have 5 parts:

```
       query: google.com
         TTL: 22
       class: IN ←(for "internet")
                    ignore this
 record type: A
record value: 172.217.13.110
```

dig TYPE domain.com

this lets you choose which DNS record to query for!

types to try: NS MX TXT CNAME A (default)

dig @8.8.8.8 domain

↑ Google DNS server

dig @server lets you pick which DNS server to query! Useful when your system DNS is misbehaving ☺

dig +trace domain

traces how the domain gets resolved, starting at the root nameservers

if you just updated DNS, dig +trace should show the new record

dig -x 172.217.13.174

makes a reverse DNS query - find which domain resolves to an IP! Same as

```
dig ptr 174.13.217.172.in-addr.arpa
```

dig +short domain

Usually dig prints lots of output! With +short, it just prints the DNS record

ping & traceroute

ping checks if you can reach a host and how long the host took to reply

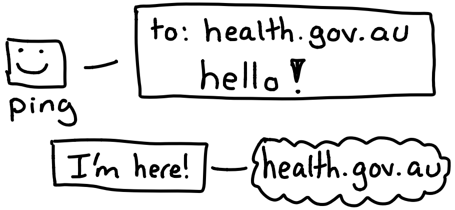
```
$ ping health.gov.au
```

output:

```
... time=253ms ...
```

Australia is 17,000 km from me. at the speed of light it's still far!

ping works by sending an ICMP packet and waiting for a reply

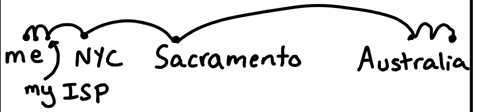


myth: if a host doesn't reply to ping, that means it's down

Some hosts never respond to ICMP packets. This is why traceroute shows "..." sometimes.

hello!
ping
not listening!!
host

traceroute tells you the path a packet takes to get to a destination



example traceroute

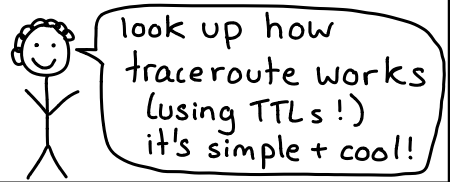
```
$ traceroute health.gov.au
1: 192.168.1.1  3ms          ← router
2: ...yul.ebox.ca 12 ms     ←ISP
          ...
8:  NYC4.ALTER.NET  24 ms
9:  SAC1.ALTER.NET  97 ms
16: health.gov.au  253ms
```

crossing the US takes time

here the packet crossed the USA! from NYC -> Sacramento!

mtr

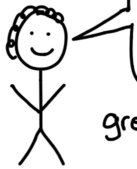
like traceroute, but nicer output! try it!



look up how traceroute works (using TTLs!) it's simple + cool!

curl

curl



great for testing APIs!

```
$ curl wizardzines.com
```

-H

is for header

good for POST requests to JSON APIs:

```
-H "Content-Type: application/json"
```

allow compressed response:

```
-H "Accept-Encoding: gzip"
```

--data

to POST data!

```
--data '{"name": "julia"}'

--data @filename.json
```

@ reads the data to send from a file

-L

follow 3xx redirects

-i

show response headers

-v

show request headers & more

-I

show only response headers (makes a HEAD request)

-k

insecure: don't verify SSL certificates

-X POST

send a POST request instead of a GET (-X PUT etc works too)

--connect-to ::IP

or hostname

send request to IP instead. use before changing DNS to a new IP

★ copy as cURL ★

Have something in your browser you want to download from the command line?

In Firefox / Chrome / Safari:

Developer Tools
→ Network tab
→ right click on the request
→ copy as curl

(can have sensitive info in cookies!)

nmap

nmap lets you explore a network

which ports are open?

which hosts are up?

security people use it a lot!

find which hosts are up

```
$ nmap -sn 192.168.1.0/24
```

↑ my home network

-sn means "ping scan" (not -s + -n, it's -sn)

just finds hosts by pinging every one, doesn't port scan

aggressive scan

```
nmap -v -A scanme.nmap.org
```

↙ aggressive

port, server version, even OS

-Pn

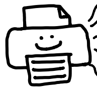
skip doing a ping scan and assume every host is up. good if hosts block ping (lots do!)

fast port scan

```
$ nmap -sS -F 192.168.1.0/24
```

just sends a SYN packet to check if each port is open.

I found out which ports my printer has open!



80	http
443	https
515	printer
631	ipp
9100	jetdirect

-F

scan less ports: just the most common ones

-T4 or -T5

scan faster by timing out more quickly

check TLS version and ciphers

check if your server still supports old TLS versions

```
$ nmap
  --script ssl-enum-ciphers
  -p 443 wizardzines.com
```

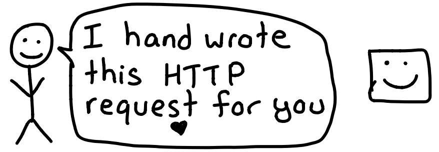
list all scripts with:

```
$ nmap --script-help '*'
```

netcat

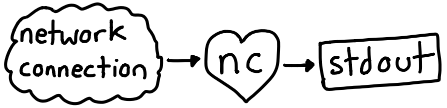
nc

lets you create TCP (or UDP) connections from the command line



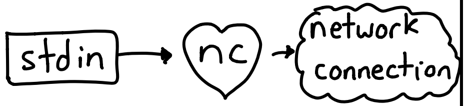
nc -l PORT

start a server! this listens on PORT and prints everything received



nc IP PORT

be a client! opens a TCP connection to IP:PORT. (to send UDP use -u)



make HTTP requests by hand

```
$ printf 'GET / HTTP/
   1.1\r\nHost:
   example.com\r\n\r\n'
  | nc example.com 80
```

all one line

type in any weird HTTP request you want! ☺

send files

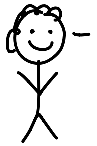
want to send a 100 GB file to someone on the same wifi network? easy!

receiver:

```
$ nc -l 8080 > file
```

sender:

```
$ cat file.txt | nc YOUR_IP 8080
```


I ♥ this trick! It works even if you're disconnected from the internet!

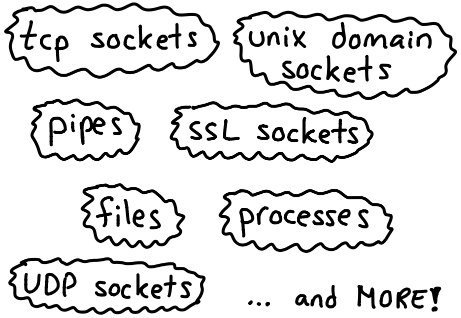
socat

socat lets you proxy basically any 2 things

the basic syntax:

```
socat THING1 THING2
```

socat supports



... and MORE!

order doesn't matter

```
socat THING1 THING2
```

is the same as

```
socat THING2 THING1
```

expose a unix domain socket on port 1337

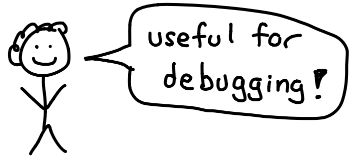
```
socat TCP-LISTEN:1337
      UNIX-CONNECT:/path
```

proxy from local HTTP port to remote server

```
socat TCP-LISTEN:1337
      TCP:domain.com:80
```

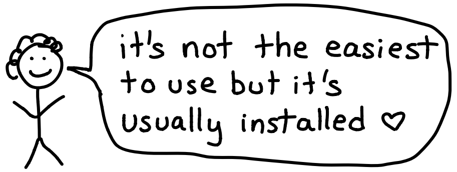
-v

write all transferred data to stderr



tcpdump

tcpdump lets you view network packets being sent & received

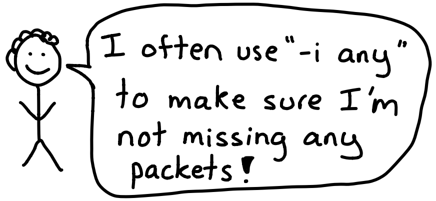


-n

don't try to resolve IP addresses/ports to DNS/port names. makes it run faster.

-i wlan0

Which network interface to capture packets on


I often use "-i any" to make sure I'm not missing any packets!

-w file.pcap

Write packets to a file for later analysis with tcpdump/tshark/wireshark/another tool

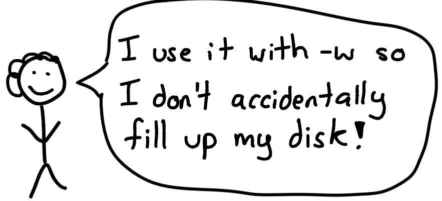
pcap is for "packet capture"

-A

print packet contents, not just headers. Nice if you want to quickly see what a few packets contain.

-c 100000

Only capture a limited count of packets



BPF cheat sheet

Berkeley Packet Filter

a small language you can use to filter which packets tcpdump and ngrep capture

Use it like this:

```
$ tcpdump [your bpf here]
$ ngrep [your bpf here]
```

host

Filter based on the source or destination IP address

use domain or IP

```
src host google.com
dst host 192.168.1.1
host 127.0.0.1
```

same as "src or dst host"

port

```
src port 53
port 80
```

again, same as "src or dst port"

less / greater

Packet length!

```
less 80
greater 200
```

and / or / not

```
host 127.0.0.1 and port 80

udp and port 53

(port 53 or port 99) and
 not host 127.0.0.1
```

PROTOCOL [INDEX]

filter based on a specific byte in a packet

IP packets with options:

bitwise and

```
ip[0] & 0xF == 5
```

DNS SERVFAIL responses:

```
udp[11] & 0xF > 0
```

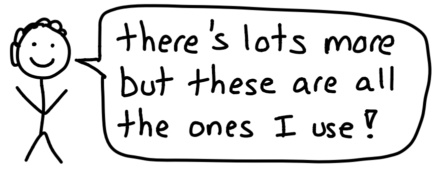
SYN packets:

```
tcp[tcpflags] == tcp-syn
```

tcp / udp / icmp

IPv4 → ip / ip6

only show packets using that protocol



tshark

♥ Wireshark ♥ is an amazing graphical packet analysis tool

tshark is the command line version of Wireshark

it can do 100x more things than tcpdump ♥

-Y

filter which packets are captured

```
tshark -Y
'http.request.method == "GET"'
```

↑ uses Wireshark's SUPER POWERFUL filter language

-d

is for "decode as"

tells tshark what protocol to interpret a port as

Example: 8888 is often HTTP!

```
$ tshark
  -d tcp.port==8888,http
```

-T FORMAT

Output format. My favourites:

- ★ json
- ★ fields: csv/tsv } for these you can specify which fields you want with -e
- ★ text: default summary

-e

Which fields to output. Ex:

```
$ tshark -T fields
  -e http.request.method
  -e http.request.uri
  -e ip.dst
```

↖ supports WAY more protocols than HTTP

```
GET /foo 92.183.216.34
POST /bar 10.23.38.132
```

-r file.pcap

analyze packets from a file instead of the network

-w

← same as tcpdump's

Write captured packets to a file. If -w file.pcap has permission issues, try:

```
tshark -w - > file.pcap
```

ngrep

like grep for your network

```
$ sudo ngrep GET
```

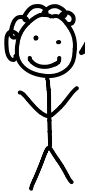
will find every plaintext HTTP GET request

ngrep syntax

```
$ ngrep
  [options]
  [regular expression]
  [BPF filter]
```

what to search packets for ↓ [regular expression]

↑ [BPF filter] same format as tcpdump uses!


I started using ngrep when I was intimidated by tcpdump and I found it easier♡

-d

is for device

which network interface to use. same as tcpdump's -i (try '-d any'!)

-W byline

prints line breaks as line breaks, not "\n". Nice when looking at HTTP requests

-I file.pcap
-O file.pcap

read/write packets from/to a pcap file

openssl

openssl is a tool for doing ★ SSL things ★
aka TLS

inspect certificates

create CSRs

sign certificates

It uses the OpenSSL library (or LibreSSL)

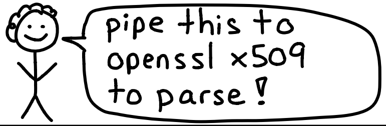
inspect a certificate

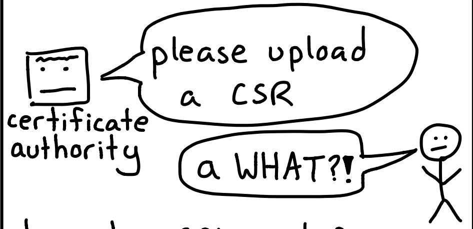
```
$ openssl x509 -in
  FILE.crt -noout -text
```

this works for files ending in .crt or .pem! Try it out: you probably have certs in `/usr/share/ca-certificates`

look at a website's certificate

```
$ openssl s_client
  -showcerts -connect
  google.com:443
```





to get a SSL cert for your website, you need to make a file called a "certificate signing request".

make a CSR

```
$ openssl req -new
  -sha256  -key FILE.key
  -out FILE.csr
```

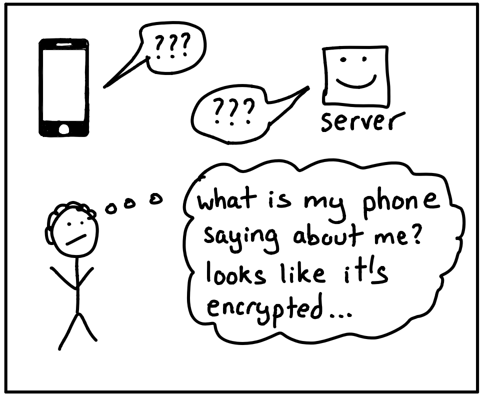
make one of these with
```
$ openssl genrsa
```

md5 / sha1 / sha256 / sha512

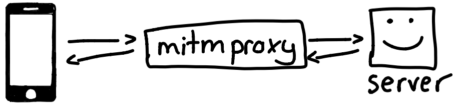
Not quite SSL but useful:
```
$ openssl md5 FILE
```
computes the md5sum of FILE. Same for other digests

```
$ openssl list -digest-commands
```
shows all supported digests.

mitmproxy


???
???
server
what is my phone saying about me? looks like it's encrypted...

mitmproxy can proxy connections from your laptop or phone and let you see the contents. It even works with encrypted connections!



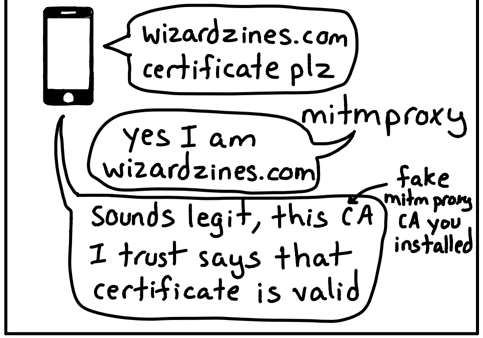
how you use it

① install mitmproxy root CA on your laptop/phone

② run mitmweb ← web UI version
on computer

③ tell the program/phone to proxy through mitmproxy

how it works


wizardzines.com certificate plz
mitmproxy
yes I am wizardzines.com
fake mitmproxy CA you installed
Sounds legit, this CA I trust says that certificate is valid

Some apps pin a cert

makes mitmproxy not work, look up "trust killer" to get around that

script it in Python

modify requests/responses arbitrarily

other similar tools

(not all are free, though)

- charles proxy
- burp suite
- fiddler

miscellaneous networking tools

stunnel
make a SSL proxy for an insecure server

hping3
make any TCP packet

wget
download files

aria2c
a fancier wget

rsync
sync files over SSH or locally

lsof
what ports are being used?

httpie
like curl but friendlier

iftop/nethogs/ntop/iptraf/nload
see what's using bandwidth

whois
is this domain registered?

ipcalc
easily see what 13.21.2.3/25 means

python3 -m http.server
serve files from a directory

nftables
new version of iptables

zenmap
GUI for nmap

p0f
identify OS of hosts connecting to you

openvpn wireguard
VPNs

tcpflow
capture and assemble TCP streams

sysctl
configure Linux kernel's network stack

ab/iperf
benchmarking tools

links
a browser in your terminal

telnet
can help debug text network protocols

ssh

♥ ssh keys ♥

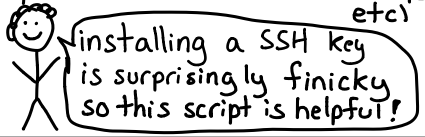
An ssh key is a secret key that lets you SSH to a machine

ssh-copy-id

This script installs your SSH key on a host (over SSH)

```
$ ssh-copy-id user@host
```

(puts it in .ssh/authorized_keys etc)


installing a SSH key is surprisingly finicky so this script is helpful!

★ port forwarding ★

```
ssh user@host.com -NfL
3333:localhost:8888
```

local port

remote port

Lets you view a remote server that's not on the internet in your browser.

just run 1 command

```
$ ssh user@host uname -a
```

runs this command & exits

~.

<Enter>~. closes the SSH connection. Useful if it's hanging!

.ssh/config

Lets you set, per host:

- username to use
- SSH key to use
- an alias!

so you can type `$ ssh ALIAS` instead of `ssh user@verylongdomain.com`

ssh-agent

remembers your SSH key passphrase so you don't have to keep typing it

mosh

ssh alternative: keeps the connection open if you disconnect + reconnect later

ip

ip

Linux only

lets you view + change network configuration.

```
$ ip OBJECT COMMAND
```

↑ addr, link neigh, etc

↑ add, show, delete, etc

ip addr list

shows ip addresses of your devices. Look for something like this:

```
2: eth0:
    link/ether 3c:97...
    inet 192.168.1.170/24
```

ip route list

displays the route table.

```
default via 192.168.1.1   ↙ my router
169.240.0.0/16 dev docker0
...
```

to see all route tables:

```
$ ip route list table all
```

change your MAC address

good for cafés with time limits 😈

```
$ ip link set wlan0 down
$ ip link set eth0 address
  3c:a9:f4:d1:00:32
$ ip link set wlan0 up
$ service network-manager
  restart  ← or whatever you use
```

ip link

network devices! (like eth0)

ip neigh

view/edit the ARP table

ip xfrm

is for IPsec

ip route get IP

what route will packets with $IP take?

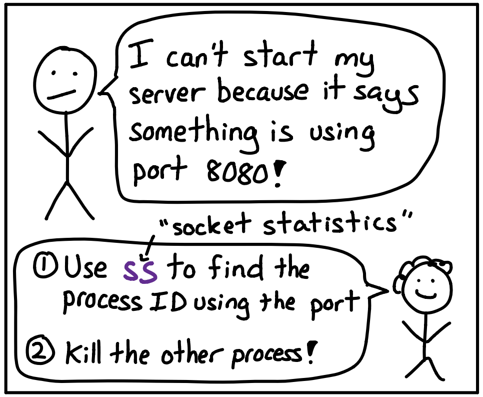
--color

pretty colourful output!

--brief

show a summary

ss


I can't start my server because it says something is using port 8080!
"socket statistics"
① Use ss to find the process ID using the port
② Kill the other process!

★ tuna, please! ★

```
$ ss -tunapl
```

the 'a' here doesn't do anything

This is my favourite way to use ss! It shows all the running servers

-n

use numeric ports (80 not http)

-p

show PIDs using the socket

TONS of information

-i -m -o

which sockets ss shows

listening or connections?

non-listening / established

default :	connections
-l :	listening
-a :	both

which protocols?

default :	all
-t :	TCP
-u :	UDP
-x :	unix domain sockets

netstat

netstat -tunapl and ss -tunapl do the same thing

netstat is older and more complicated. If you're learning now, I'd recommend ss!

iptables

iptables lets you create rules to match network packets and accept/drop/modify them

It's used for firewalls and NAT

tables have chains chains have rules

tables: `filter, nat, mangle, raw, security`

chains: `INPUT, FORWARD, PREROUTING,` etc

rules: like `-s 10.0.0.0/8 -j DROP`

iptables-save

This prints out all iptables rules. You can restore them with

`iptables-restore`

but it's also the easiest way to view all rules!

-j TARGET

Every iptables rule has a target (what to do with matching packets). Options:

→ `ACCEPT, DROP, RETURN`

→ the name of an iptables chain

→ an extension (man iptables-extensions)
Popular: `DNAT, LOG, MASQUERADE`

tables have different chains

filter:	`INPUT, OUTPUT, FORWARD`
mangle:	`INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING`
nat:	`OUTPUT, PREROUTING, POSTROUTING`

It helps to know when packets get processed by a given table/chain (eg locally generated packets go through `filter` and `OUTPUT`)

you can match lots of packet attributes

-s: src ip	-p: tcp/udp
-d: dst ip	-i: network interface

-m: lots of things! (bpf rules! cgroups! ICMP type! cpu! conntrack state! more!)

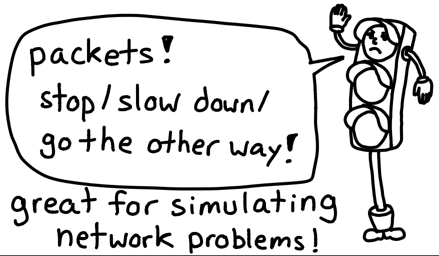
For more, run:

```
$ man iptables-extensions
```

tc

tc

is for "traffic control"



great for simulating network problems!

make your internet slow

```
$ sudo tc qdisc add dev
  wlp3s0 root netem
  delay 500ms
```

← delay packets by 500ms

and fast again.

```
$ sudo tc qdisc del dev
  wlp3s0 root netem
```

netem rules

netem ("network emulator") is a part of tc that lets you:

{drop} {duplicate} {delay} {corrupt}

packets. See the man page:

```
$ man netem
```

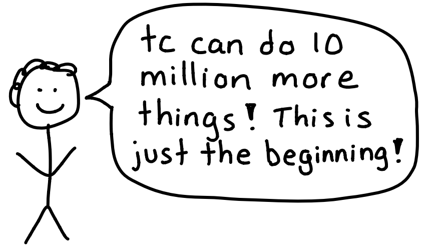
make your brother's internet slow

Have a Linux router? You can configure tc on it to make your brother's internet slower than yours

google: "tc QoS" for a start

show current tc settings

```
$ tc qdisc show
$ tc class show dev DEV
$ tc filter show dev DEV
```


tc can do 10 million more things! This is just the beginning!

conntrack

conntrack

not a command line tool:
it's a Linux kernel system for tracking TCP/UDP connections.

It's a kernel module called nf_conntrack

conntrack is used for:

- NAT (in a router!)
- firewalls (eg only allow outbound connections)

You control it with iptables rules.

conntrack has a table of every connection

Each entry contains:

- src + dest IP
- src + dest ports
- the connection state (eg TIME_WAIT)

how to enable conntrack

enable:

```
$ sudo modprobe nf_conntrack
```

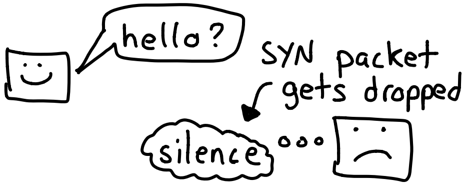
check if it's enabled:

```
$ lsmod | grep conntrack
```

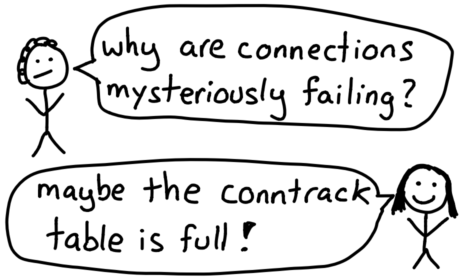
change table size with the sysctl

```
net.netfilter.nf_conntrack_max
```

if the conntrack table gets full, no new connections can start

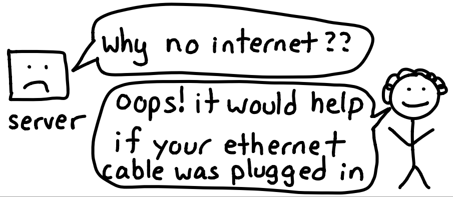


moral: be careful about enabling conntrack!


why are connections mysteriously failing?
maybe the conntrack table is full!

ethtool

ethtool is for people who need to manage physical networks



ethtool eth0

name of network interface

this tells you:

- is it even connected? ("link detected")
- speed
- lots more

--show-offload / --offload

your network card can do a lot for you! Like computing checksums. This is called "offloading".

This lets you see/change configured offloads.

--identify INTERFACE

blink the light on the ethernet port. good if you have multiple ports! and cute ♥

-s

change speed/duplex/other settings of an interface

```
$ ethtool -s eth0 speed 100
```

iw dev wlan0 link

ethtool is mostly for Ethernet.

To see the speed (and more) of a wireless connection, use iw.

-S INTERFACE

show statistics like bytes sent. works for wifi interfaces too.

-i INTERFACE

show firmware info