ADVANCED CLOUD COMPUTING

CONTAINERS

DEEP DIVE, PART 2

NAMESPACES

PID NS

PID ns **isolate the process ID** number space

processes in different PID ns can have the same PID.

functionalities:

- suspending/resuming
- migrating the container

There is a mapping between pid inside and outside a container

IN PRACTICE

Outside the container

```
$ pstree -N pid -p
# Main PID ns
[4026531836]
alacritty(571899)-+-zsh(571908)---pstree(573825)
alacritty(573585)-+-zsh(573594)---podman(573756)-+-{podman}(57

# container pid ns form the outside
[4026533992]
bash(573787)
```

In the container

```
[root@07fa0660c9ea /]# pstree -N pid -p
# container PID ns
[4026533992]
# bash(573787) maps to one
bash(1)───pstree(37)
```

PID ONE

PIDs in a new PID ns start at 1, like a standalone system

PID ONE

IN GENERAL

1. kernel is loaded
2. runs a user-space program (first process): PID 1.

this program is often `systemd`. It:

- is the parent of all processes
- start daemons
- manages runlevel/targets
- handle shutdown/reboots/critical events

PID ONE

IN CONTAINERS

- Running the primary application (eg. nginx)
- acting as the parent for all other processes (eg. nginx worker)
- cleaning up resources (e.g., zombie processes).

mantra "one service per container"

CLEANING UP

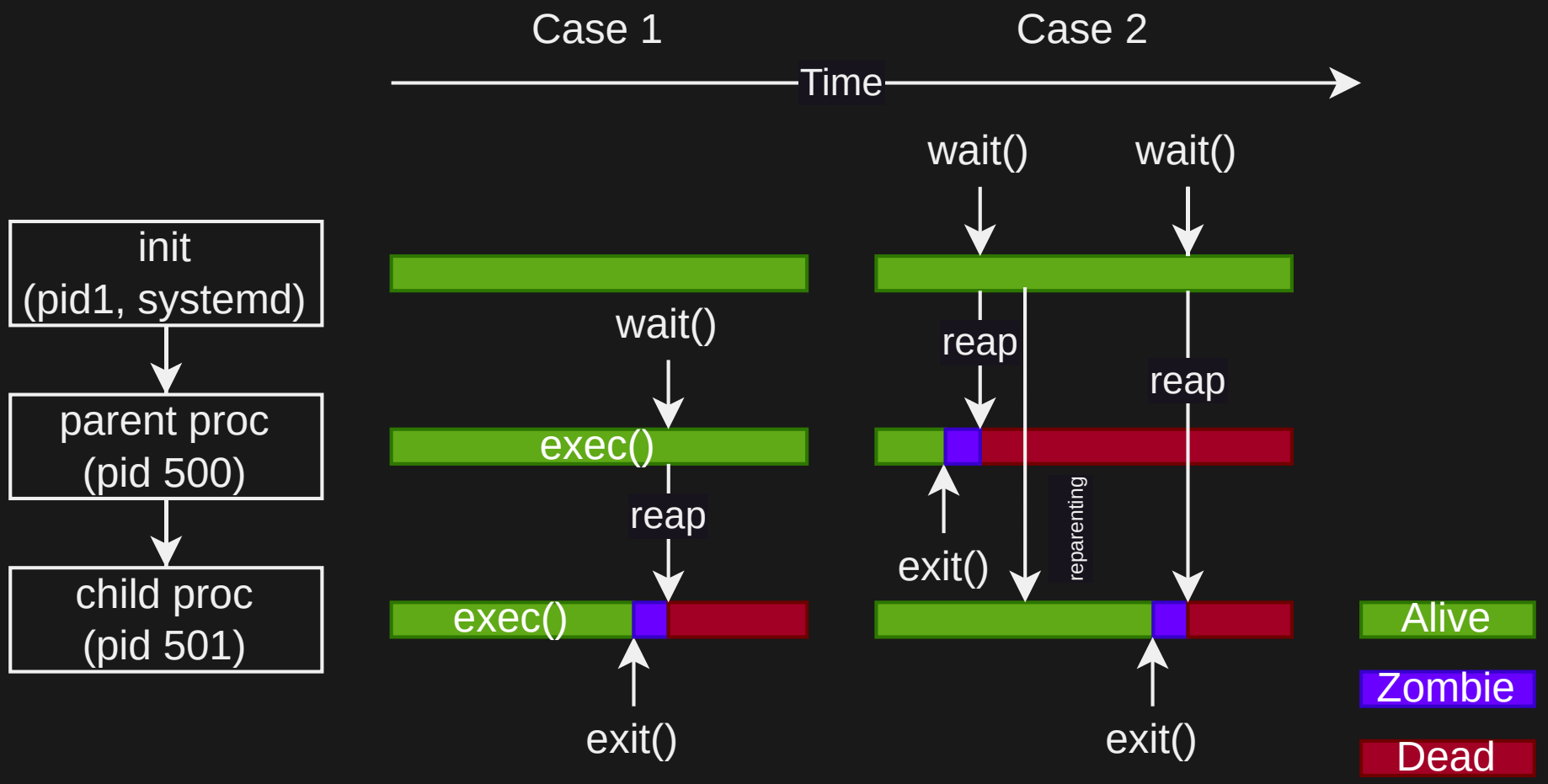
IN WHAT SENSE?

SIGNALS FOR IPC COMMUNICATION

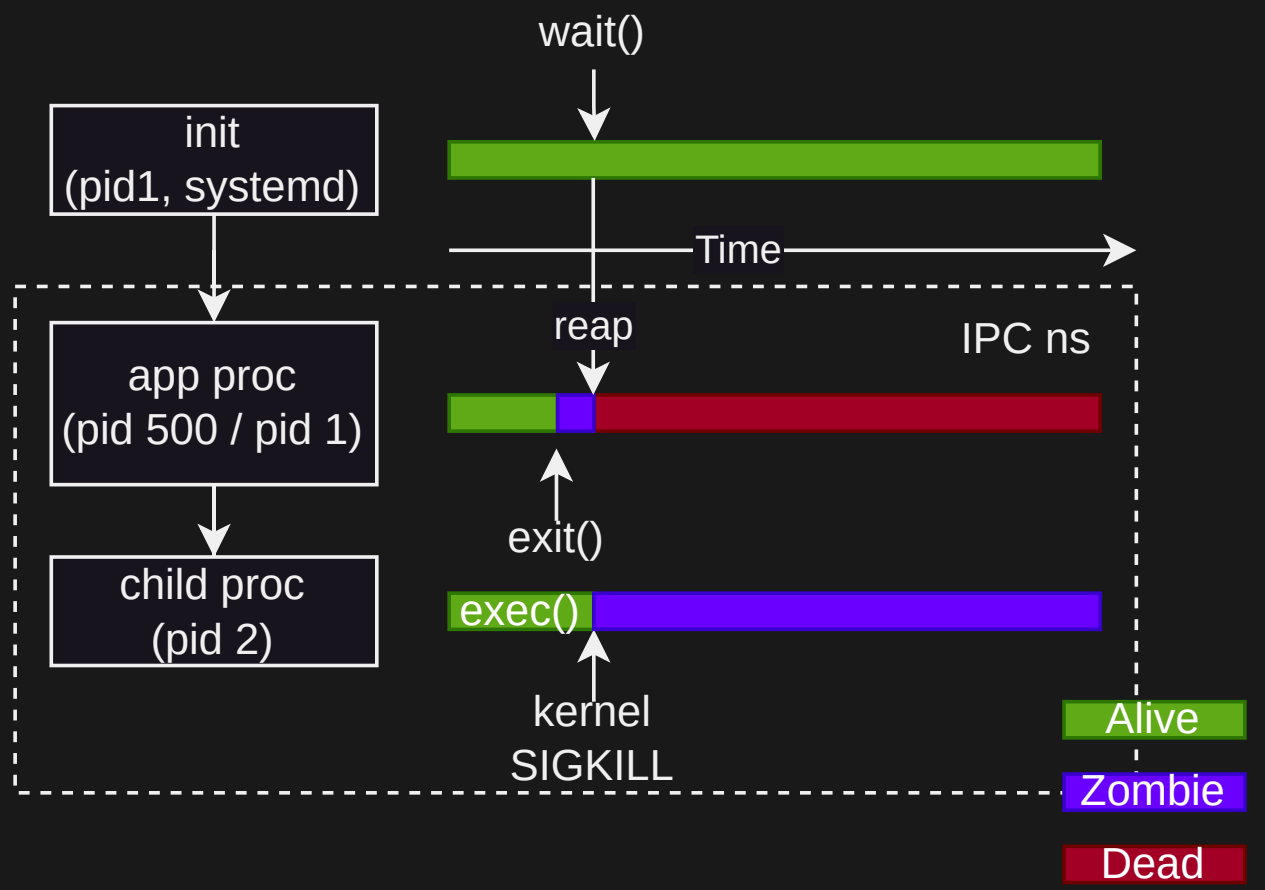
Signal	Value[x86]	Action	comment
SIGHUP	1	Term	Hangup (*)
SIGINT	2	Term	Interrupt from keyboard
SIGKILL	9	Term	Kill signal
SIGTERM	15	Term	Termination signal
SIGCHLD	17	Ign	Child stopped/terminated
SIGCONT	18	Cont	Continue if stopped
SIGSTOP	19	Stop	Stop process

only implemented signal are handled, exceptions: kill/stop

TYPICAL PROCESS EVOLUTION



THE CONTAINER ISSUE



the PID ns can not be destroied if it has prcesses.

THE CONTAINER ISSUE

- your app dies for whatvere reason(killed, crash)
- in this case there is no more pid 1, reparenting can not happen in the ns
- the ns will exists forever (until reboot), no longer usable

if your app is poorly written/has complex dependencies use an init system like s6

PSEUDO FILE SYSTEMS

THE /PROC EXAMPLE

The proc file system acts as an interface to internal data structures in the kernel.

It can be used to:

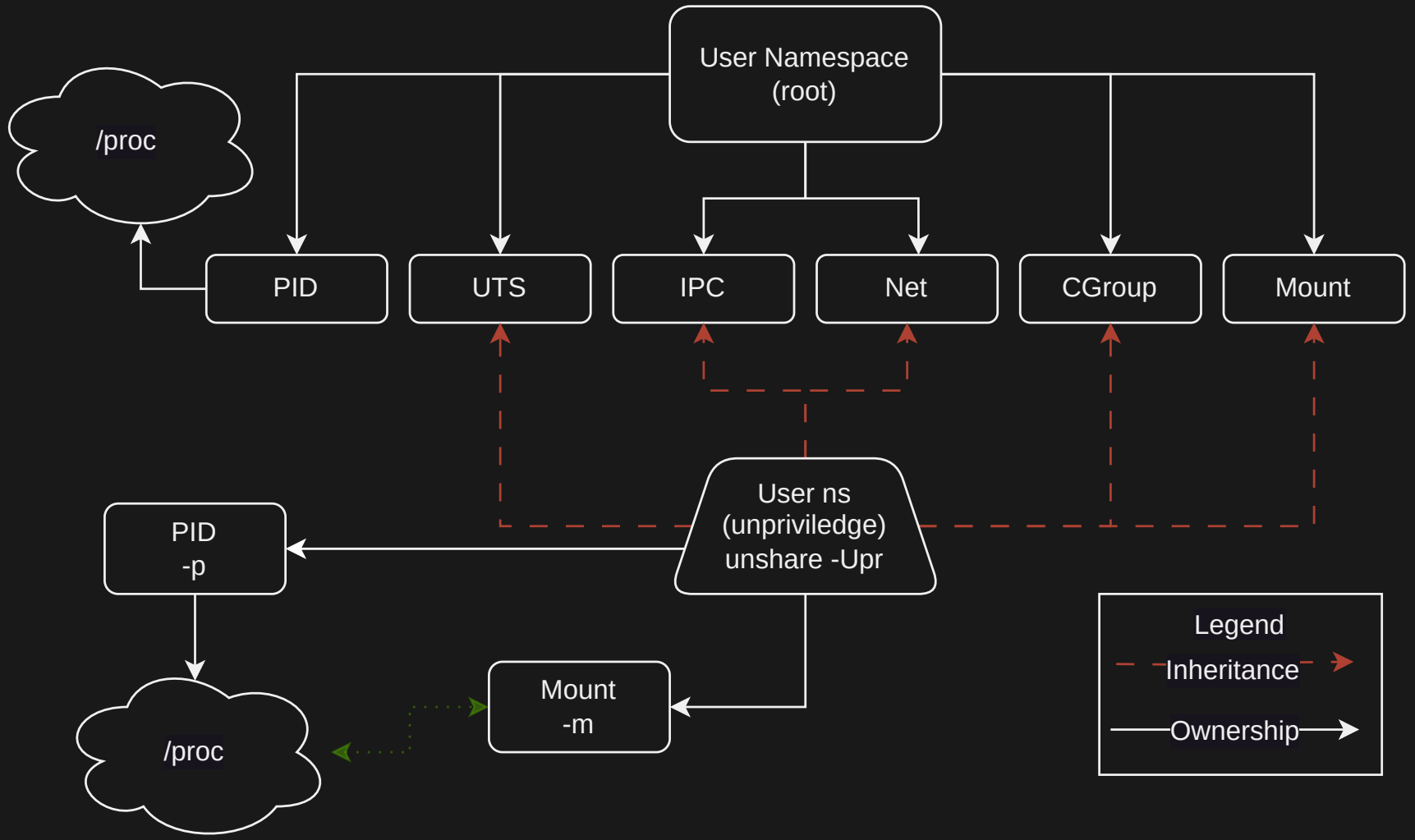
- obtain information about the system process
- change certain kernel parameters at runtime.

Now: fixing the proc mounting of last time

HANDSON

Stream online

WHAT DID WE DO?



UTS NAMESPACE

It govern the hostname, (openssl certificates)

```
➜  ~ unshare -Urmpu -f /usr/bin/bash
bash-5.2# hostname
Derinoe
bash-5.2# hostname pippo
bash-5.2# hostname
pippo
bash-5.2#
```

IPC

POSIX message queues, another way to exchage data between processes in the form of messages.

TIME

time is to play with time

```
➜  ~ unshare -Urmpu -f /usr/bin/bash
bash-5.2~ uptime
 21:03:23 up 4 days,  5:20
➜  ~ unshare -UrmpuT --boottime 2000000000 -f /usr/bin/bash
bash-5.2~ uptime
 21:04:05 up 23152 days,  8:54
```

CGROUP

Add a cgroup fs

when live migrating containers, and for security.

Cgroup	controlled quantity
`cpu.max`	Max cpu
`io.max`	Max IOps
`memory.max`	Max RAM (hard limit), proc killed at reacing
`memory.min`	min RAM (hard limit), proc killed if requirement can't be satisfied

NETWORK NAMESPACES

FIRST: NETWORK INTERFACES

KIND OF INTERFACES

- amt
- bareudp
- bond
- bond_slave
- bridge
- bridge_slave
- dsa
- dummy
- erspan
- geneve
- gre
- gretap
- gtp
- ifb
- ipip
- ipoib
- ipvlan
- ipvtap
- macsec
- macvlan
- macvtap
- netdevsim
- nlmon
- rmnet
- sit
- team
- team_slave
- vcan
- veth
- vlan
- vrf
- vti
- vxcan
- vxlan
- wwan
- ..more..

THE ONE YOU MIGH CARE ABOUT

- amt
- bareudp
- bond
- bond_slave
- bridge
- bridge_slave
- dsa
- dummy
- erspan
- geneve
- gre
- gretap
- gtp
- ifb
- ipip
- ipoib
- ipvlan
- ipvtap
- macsec
- macvlan
- macvtap
- netdevsim
- nlmon
- rmnet
- sit
- team
- team_slave
- vcan
- veth
- vlan
- vrf
- vti
- vxcan
- vxlan
- wwan
- ..more..

THE ONE YOU MUST KNOW

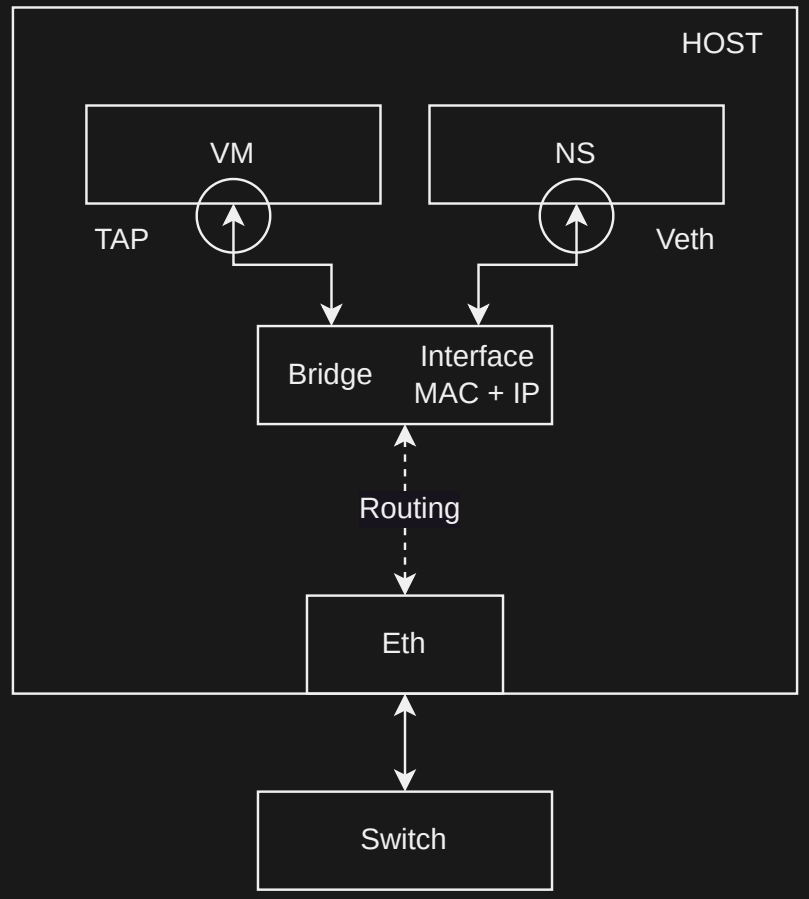
- bond
- bridge
- veth
- vlan
- vxlan

THE ONE YOU MUST KNOW

- bond
- bridge
- veth
- vlan
- vxlan

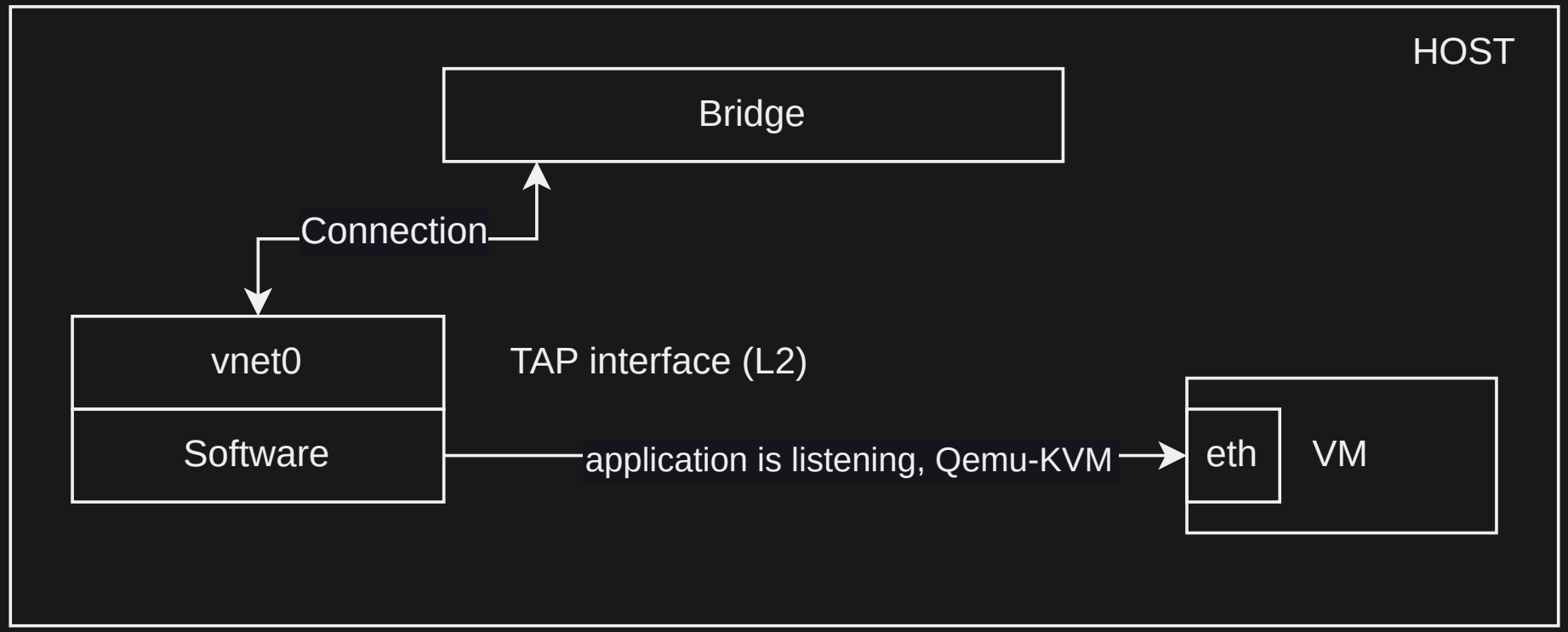
the triky tun/tap

BRIDGE



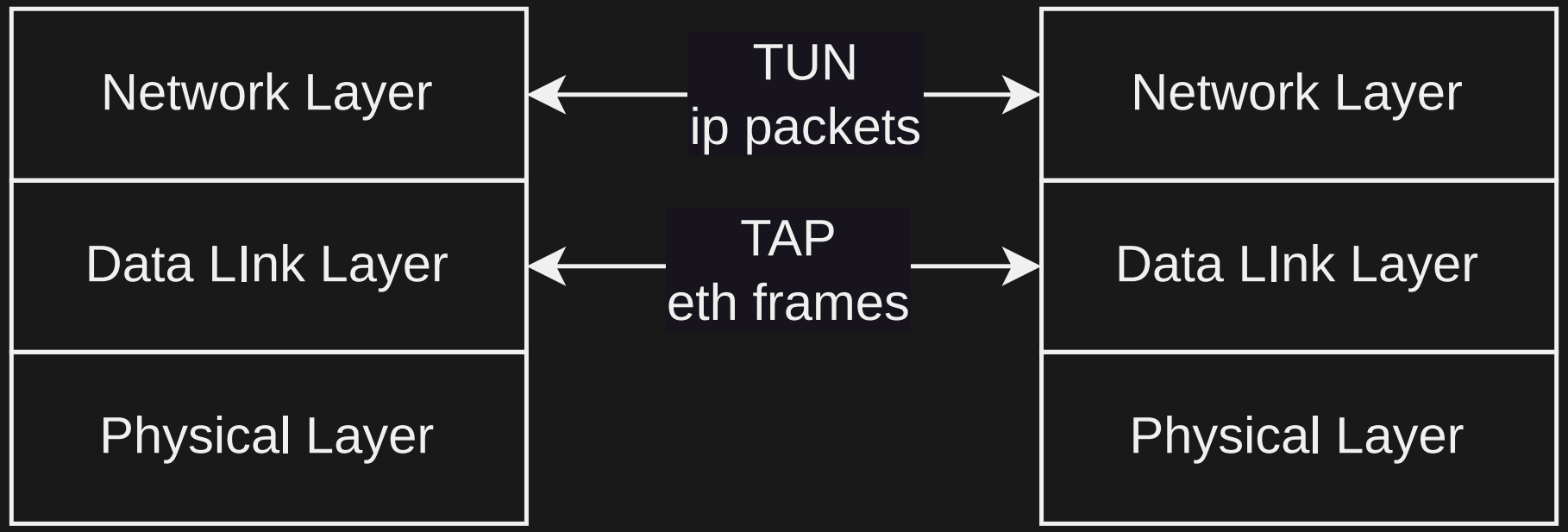
a bridge to enstablish connection between containers or VM

TAP



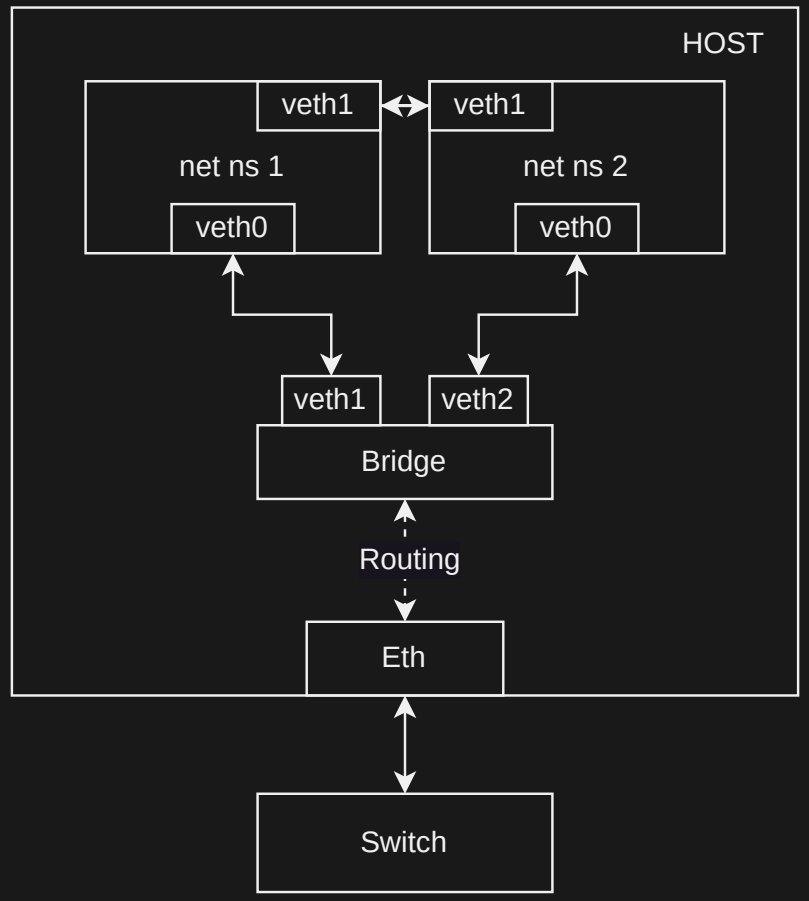
used to connect VM

TUN



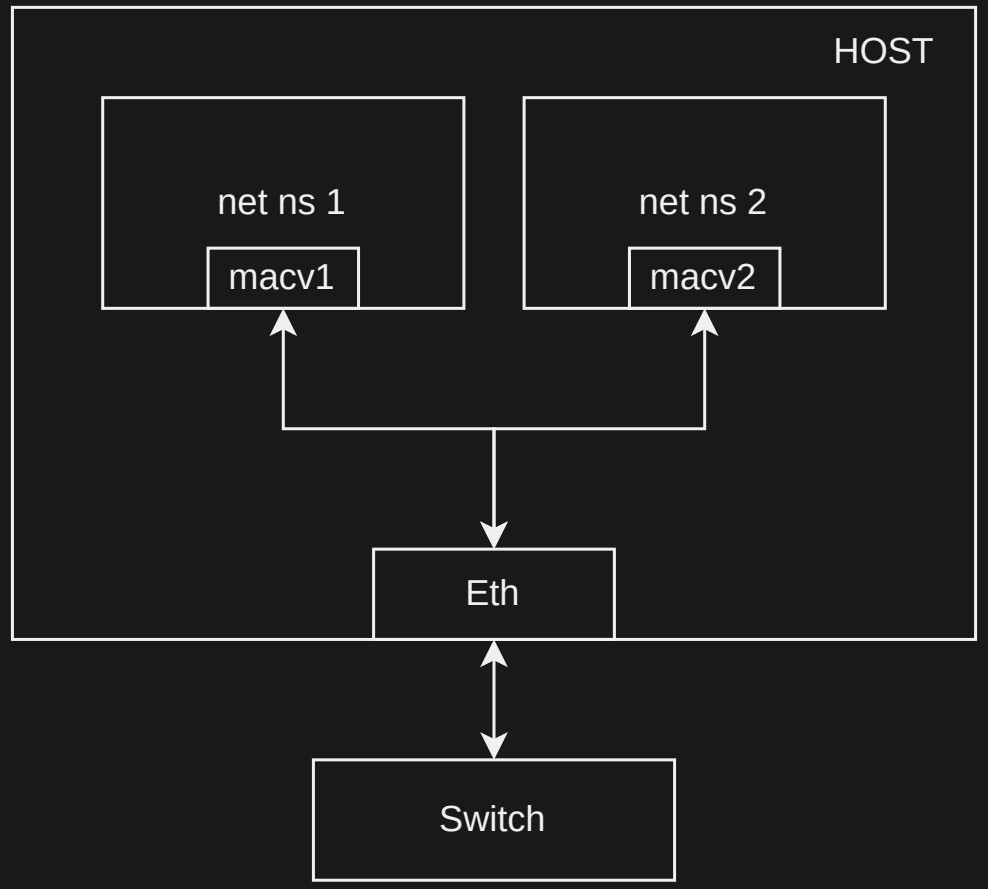
used by applications (VPN)

VETH



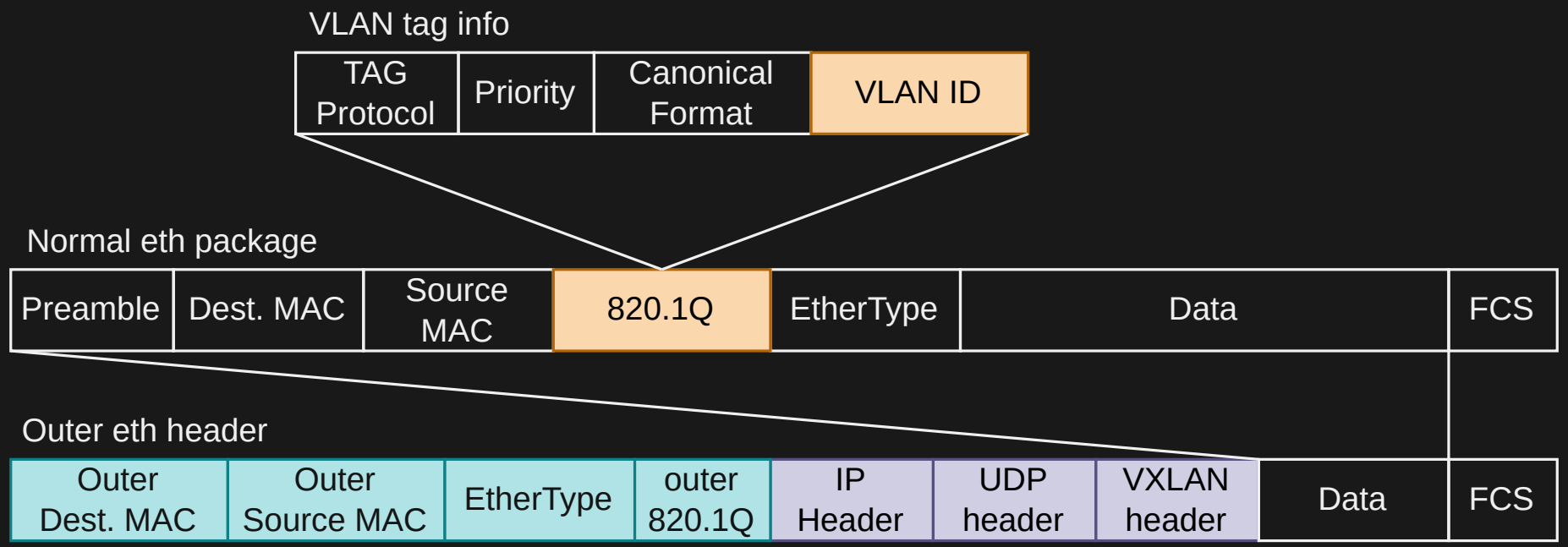
used to connect containers

MACVLAN



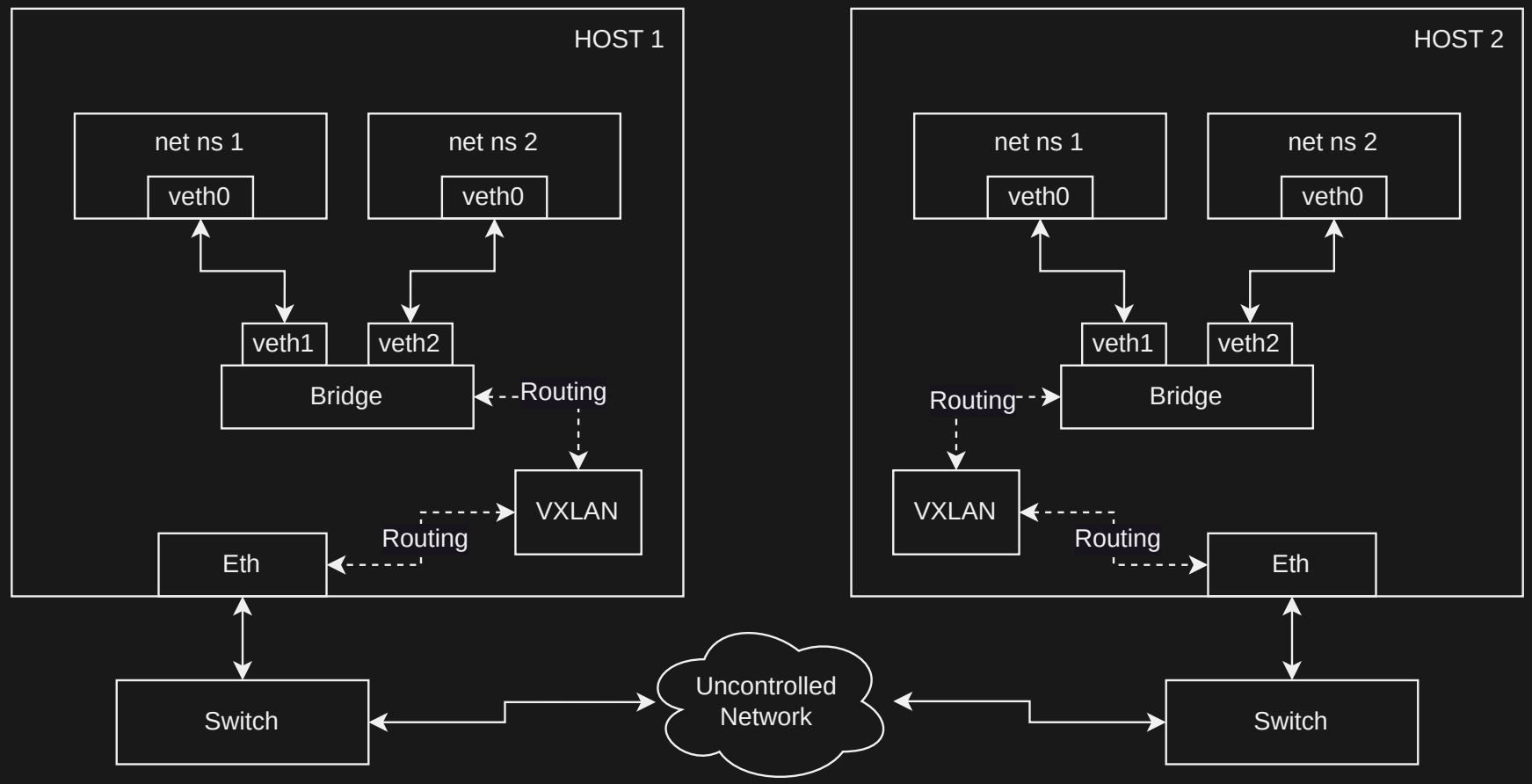
used

VLAN/VXLAN



used

VXLAN



kube CNI implementation

THE `ip addr` COMMAND

```
1: lo: <loopback>
    mtu 65536 qdisc noop state DOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00</loo
```

allow internal communication with local services that can be hosted for diagnostics, troubleshooting

eg: name resolution (DNS)

THE `ip addr` COMMAND

```
2: ptp-veth-one@ptp-veth-two: <broadcast,multicast,m-down>
    mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether f6:95:31:21:19:ea brd ff:ff:ff:ff:ff:ff</broadcast,multicast,m-down>
```

- # number
- one@two
- BROADCAST,MULTICAST
- mtu 1500
- qdisc noop
- state DOWN
- group default
- qlen 1000
- link/ether

number represent the unique interface id.

THE `ip addr` COMMAND

```
2: ptp-veth-one@ptp-veth-two: <broadcast,multicast,m-down>
    mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether f6:95:31:21:19:ea brd ff:ff:ff:ff:ff:ff</broadcast,multicast,m-down>
```

- # number
- ptp-veth-one@ptp-veth-two
- BROADCAST,MULTICAST
- mtu 1500
- qdisc noop
- state DOWN
- group default
- qlen 1000
- link/ether

is the interface name before the @ and where it goes

THE `ip addr` COMMAND

```
2: ptp-veth-one@ptp-veth-two: <broadcast,multicast,m-down>
    mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether f6:95:31:21:19:ea brd ff:ff:ff:ff:ff:ff</broadcast,multicast,m-down>
```

- # number
- one@two
- BROADCAST,MULTICAST,M-DOWN
- mtu 1500
- qdisc noop
- state DOWN
- group default
- qlen 1000
- link/ether

This interface supports broad and multicasting

THE `ip addr` COMMAND

```
2: ptp-veth-one@ptp-veth-two: <broadcast,multicast,m-down>
    mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether f6:95:31:21:19:ea brd ff:ff:ff:ff:ff:ff</broadcast,multicast,m-down>
```

- # number
- one@two
- BROADCAST,MULTICAST
- mtu 1500
- qdisc noop
- state DOWN
- group default
- qlen 1000
- link/ether

The maximum transfer unit this interface supports.

THE `ip addr` COMMAND

```
2: ptp-veth-one@ptp-veth-two: <broadcast,multicast,m-down>
    mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether f6:95:31:21:19:ea brd ff:ff:ff:ff:ff:ff</broadcast,multicast,m-down>
```

- # number
- one@two
- BROADCAST,MULTICAST
- mtu 1500
- qdisc noop
- state DOWN
- group default
- qlen 1000
- link/ether

The scheduler is using a discipline called, none.

THE ip addr COMMAND

```
2: ptp-veth-one@ptp-veth-two: <broadcast,multicast,m-down>
    mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether f6:95:31:21:19:ea brd ff:ff:ff:ff:ff:ff</broadcast,multicast,m-down>
```

- # number
- one@two
- BROADCAST,MULTICAST
- mtu 1500
- qdisc noop
- state DOWN
- group default
- qlen 1000
- link/ether

The interface is not connected. To bring it up we must issue the ip link set dev interface_name up command on both sides of the cable.

THE `ip addr` COMMAND

```
2: ptp-veth-one@ptp-veth-two: <broadcast,multicast,m-down>
    mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether f6:95:31:21:19:ea brd ff:ff:ff:ff:ff:ff</broadcast,multicast,m-down>
```

- # number
- one@two
- BROADCAST,MULTICAST
- mtu 1500
- qdisc noop
- state DOWN
- group default
- qlen 1000
- link/ether

This interface is in the "default" interface group.

THE `ip addr` COMMAND

```
2: ptp-veth-one@ptp-veth-two: <broadcast,multicast,m-down>
    mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether f6:95:31:21:19:ea brd ff:ff:ff:ff:ff:ff</broadcast,multicast,m-down>
```

- # number
- one@two
- BROADCAST,MULTICAST
- mtu 1500
- qdisc noop
- state DOWN
- group default
- qlen 1000
- link/ether

The maximum length of the transmission queue.

THE `ip addr` COMMAND

```
2: ptp-veth-one@ptp-veth-two: <broadcast,multicast,m-down>
    mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether f6:95:31:21:19:ea brd ff:ff:ff:ff:ff:ff</broadcast,multicast,m-down>
```

- # number
- one@two
- BROADCAST,MULTICAST
- mtu 1500
- qdisc noop
- state DOWN
- group default
- qlen 1000
- link/ether

The MAC address of the interface, and broadcast

SECOND: INTERFACE AND NAMESPACE

NETWORK NS

isolation of the systemnetworking resources:

- network devices,
- IPv4 and IPv6 protocol stacks,
- port numbers (sockets),
- IP routing tables,
- firewall rules,
- the /proc/net ,/sys/class/net directories

RUN FS

run directory is designed to allow applications to store the data they require. Files in this directory include process IDs, socket information, lock files, and other data that is required at runtime

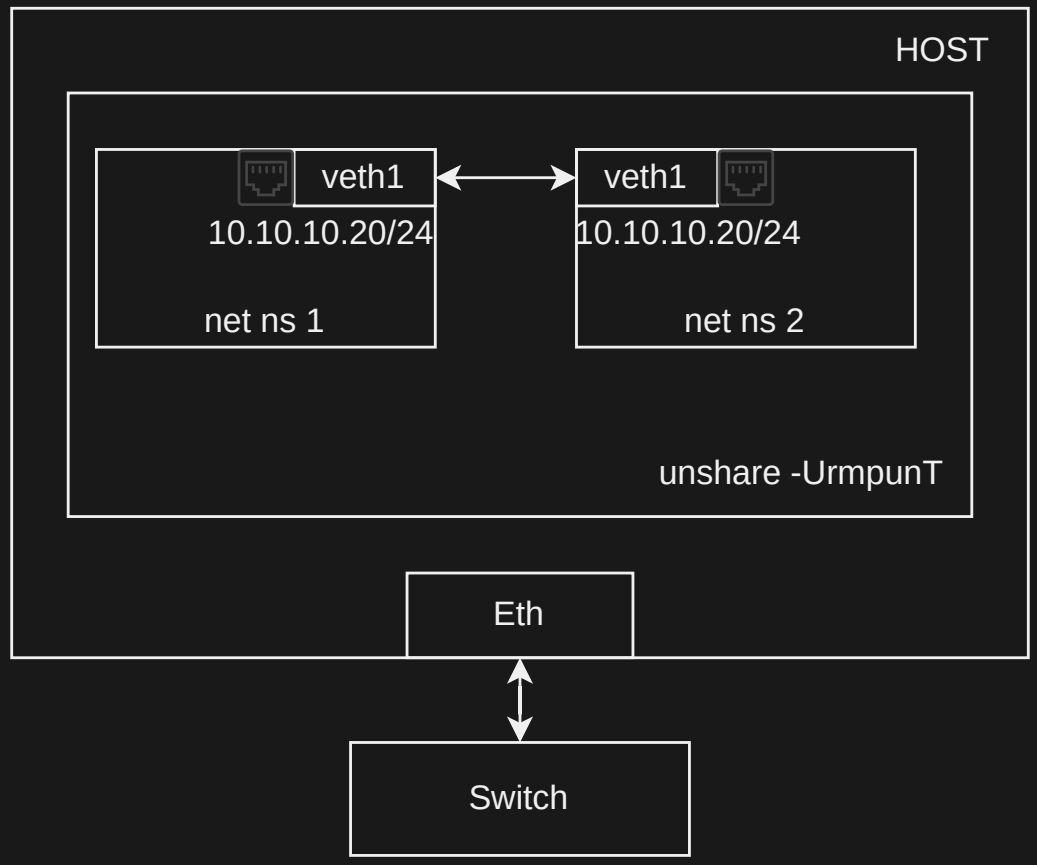
run is needed for ip to work and have the necessary privileges to create the files corresponding to the network namespaces we are creating.

EXERCISE ONE

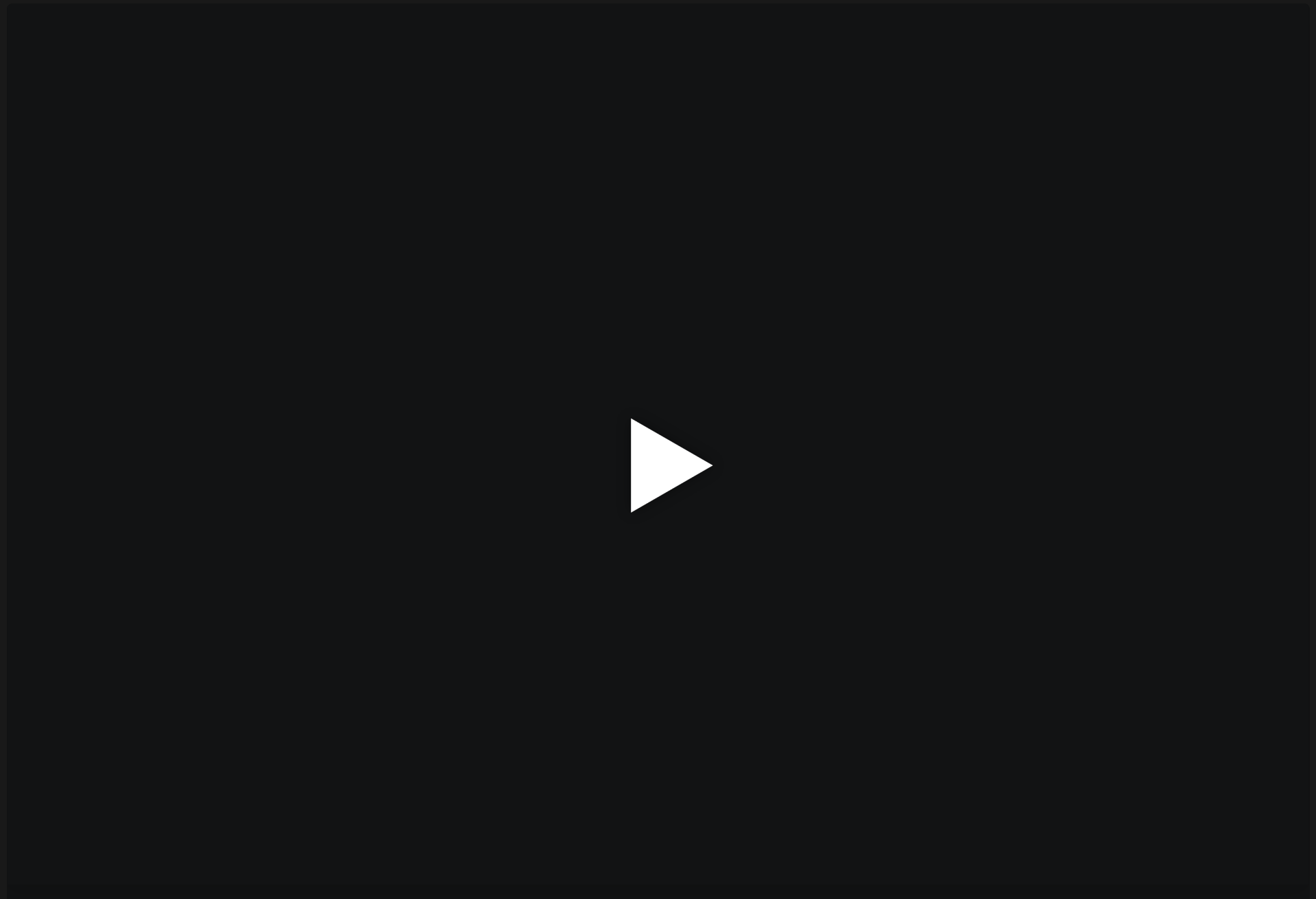
CONNECT TWO NETWORK NS DIRECTLY

EXERCISE ONE

CONNECT TWO NETWORK NS DIRECTLY



Stream online

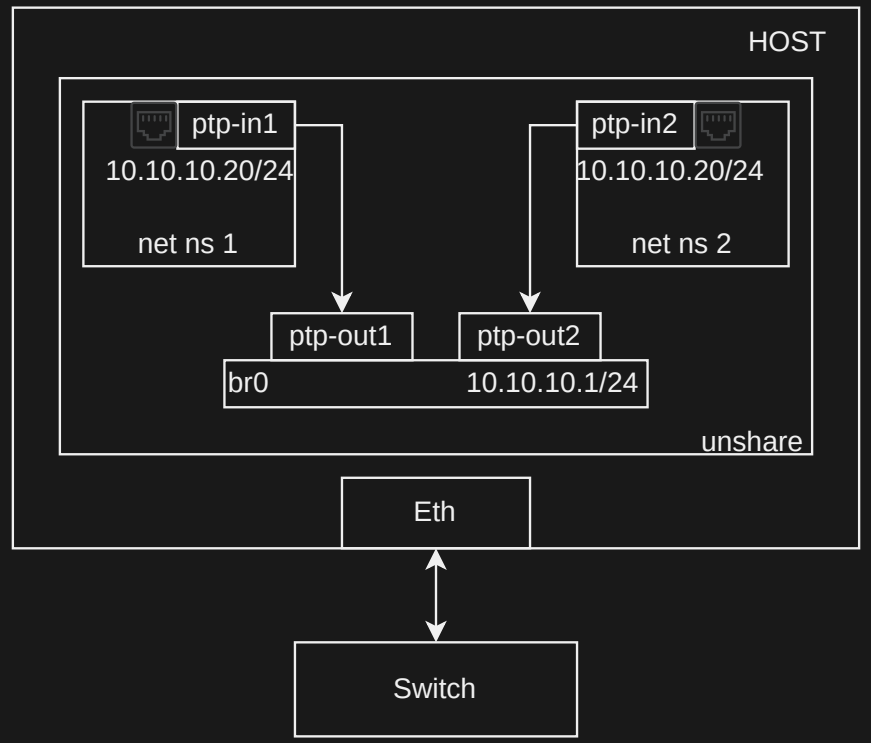


EXERCISE TWO

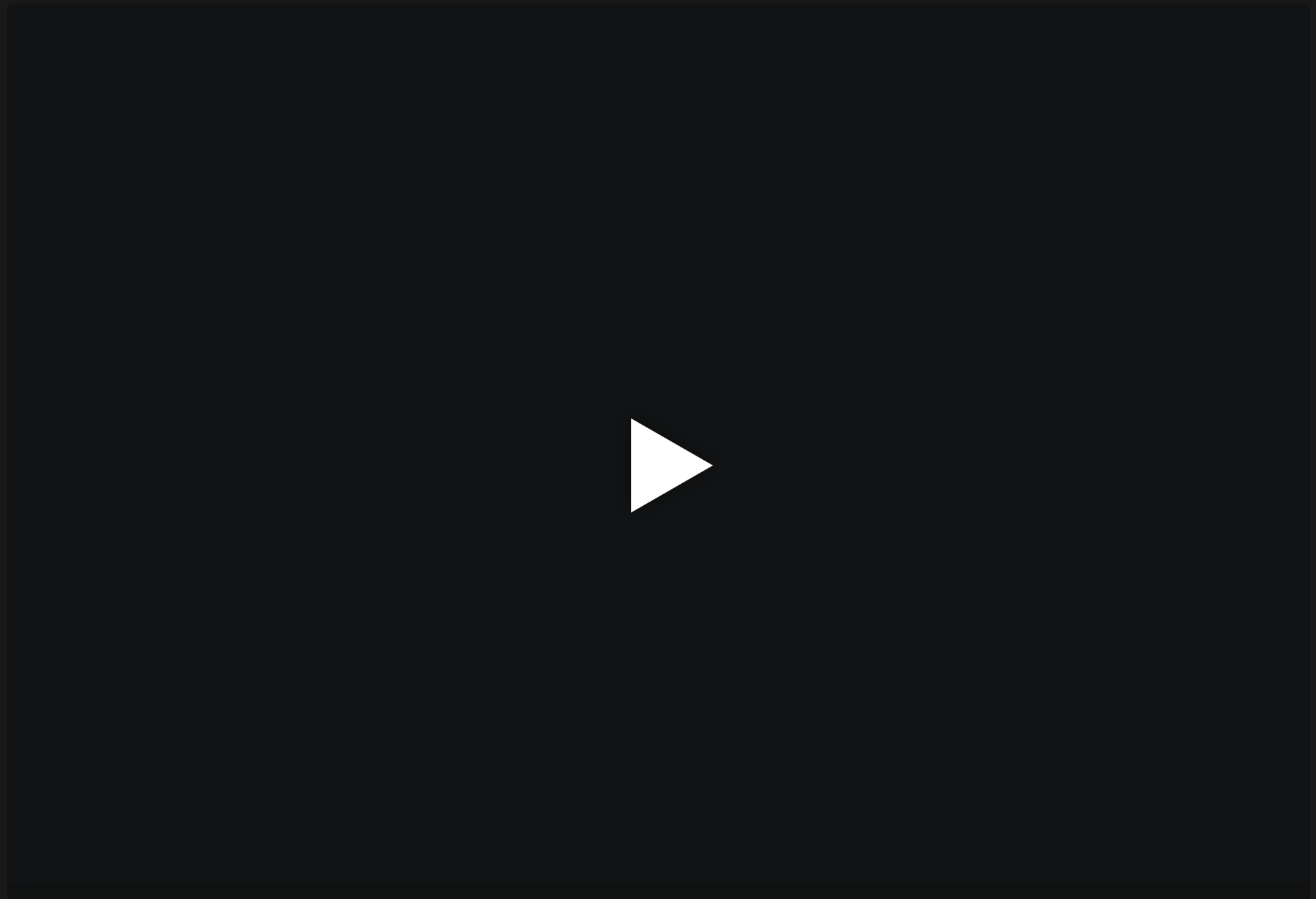
CONNECT TWO NETWORK NS INDIRECTLY

EXERCISE TWO

CONNECT TWO NETWORK NS INDIRECTLY



Stream online



INTENRENT IN NETNS

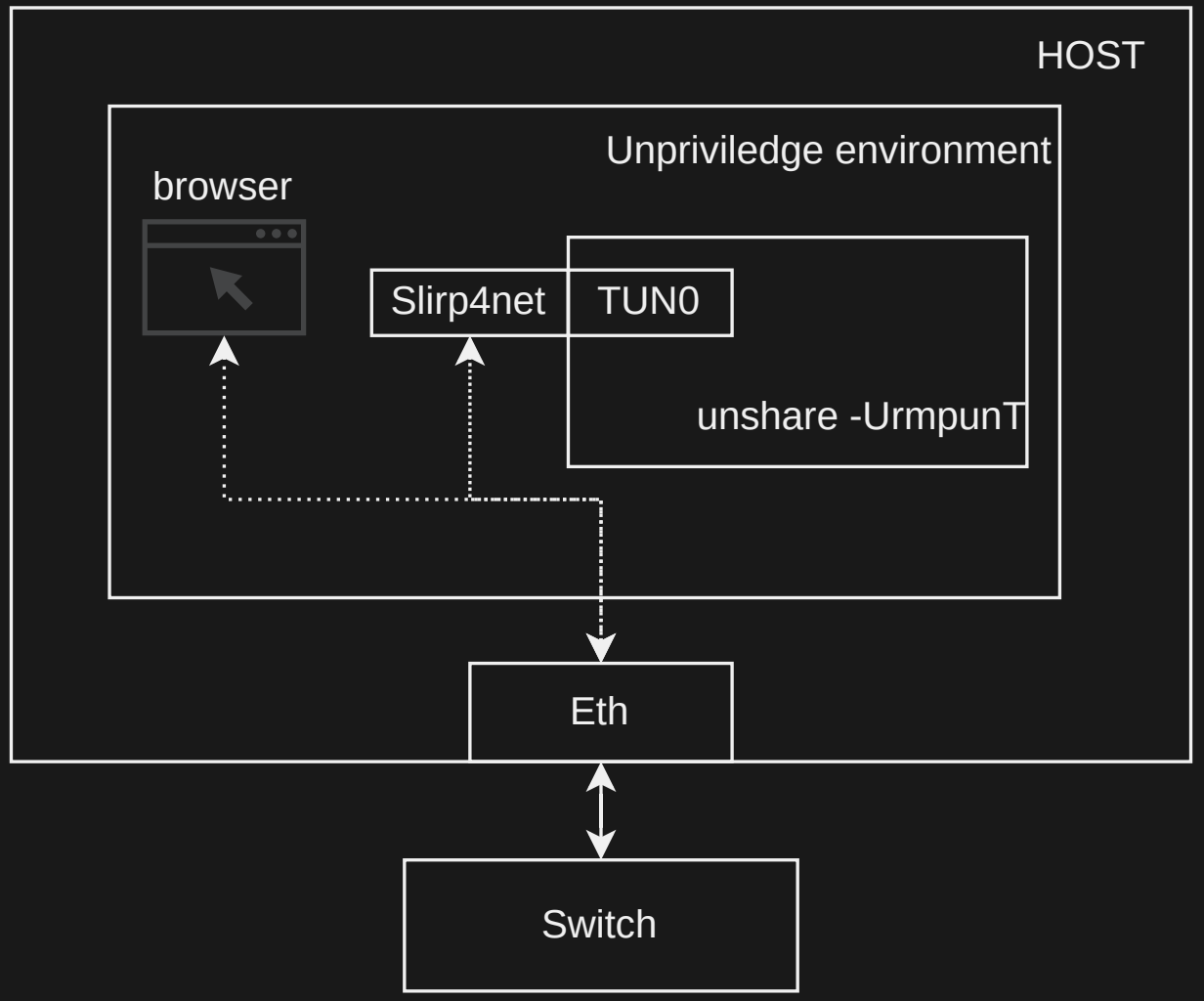
(UNPRIVILEDGE VERSION)

the traffic must be seen as if your user generated it

solutions:

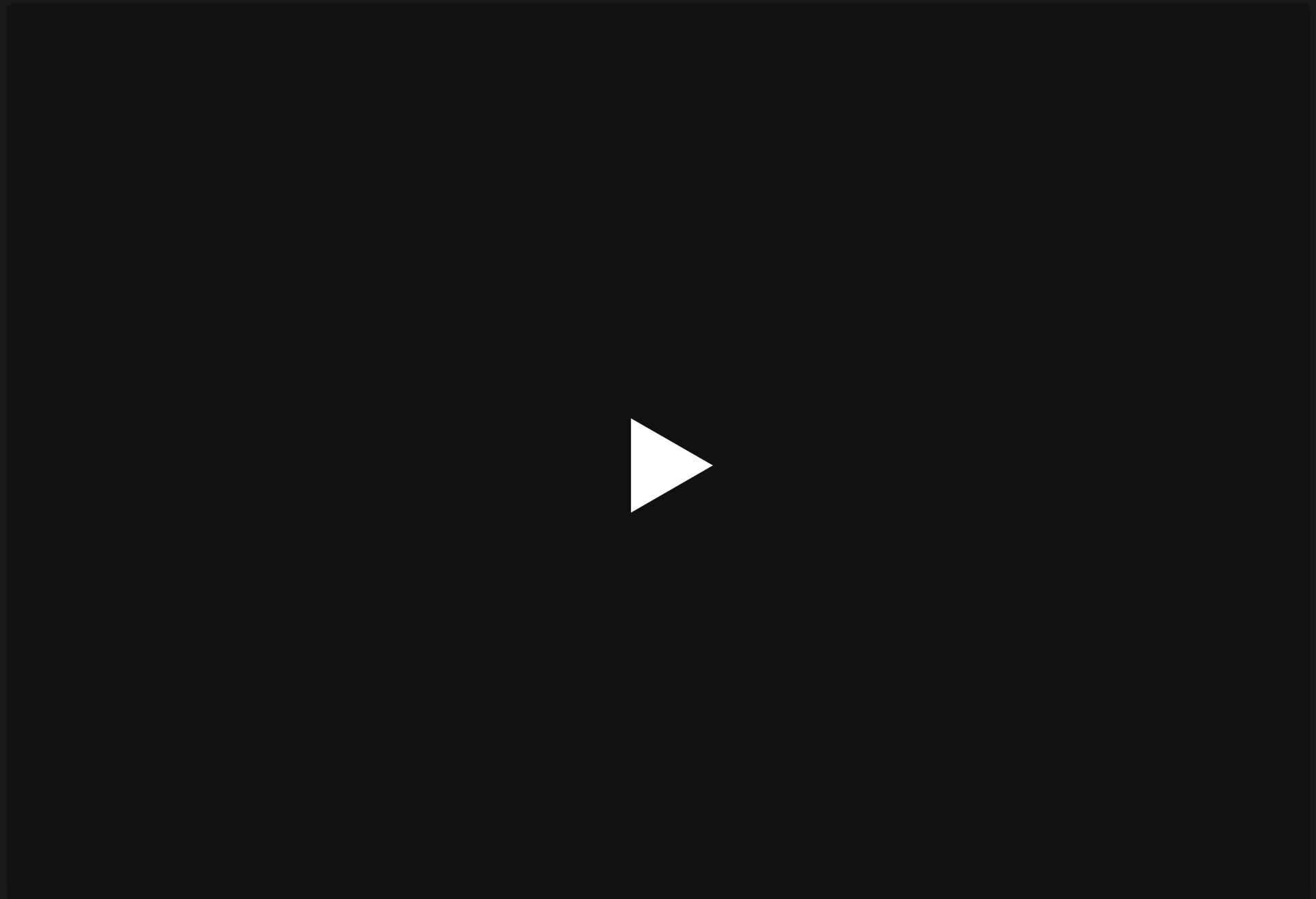
- slirp4netns
- VPNKit
- vde_plug

INTERNET IN NETNS



INTERNET IN NETNS

Stream online



UNSHARE -> OCI RUNTIME (CRUN)

substitute our unashare call with crun, the OCI runtime.