Rootless Networking

From possible to practical

David Gibson <dgibson@redhat.com>
Principal Software Engineer

Red Hat

An aside

If you saw my presentation from last Everything Open

Covering some of the same topics as Everything Open 2023 talk

- Different perspective:
  - Previous talk was historical
  - This one is practical

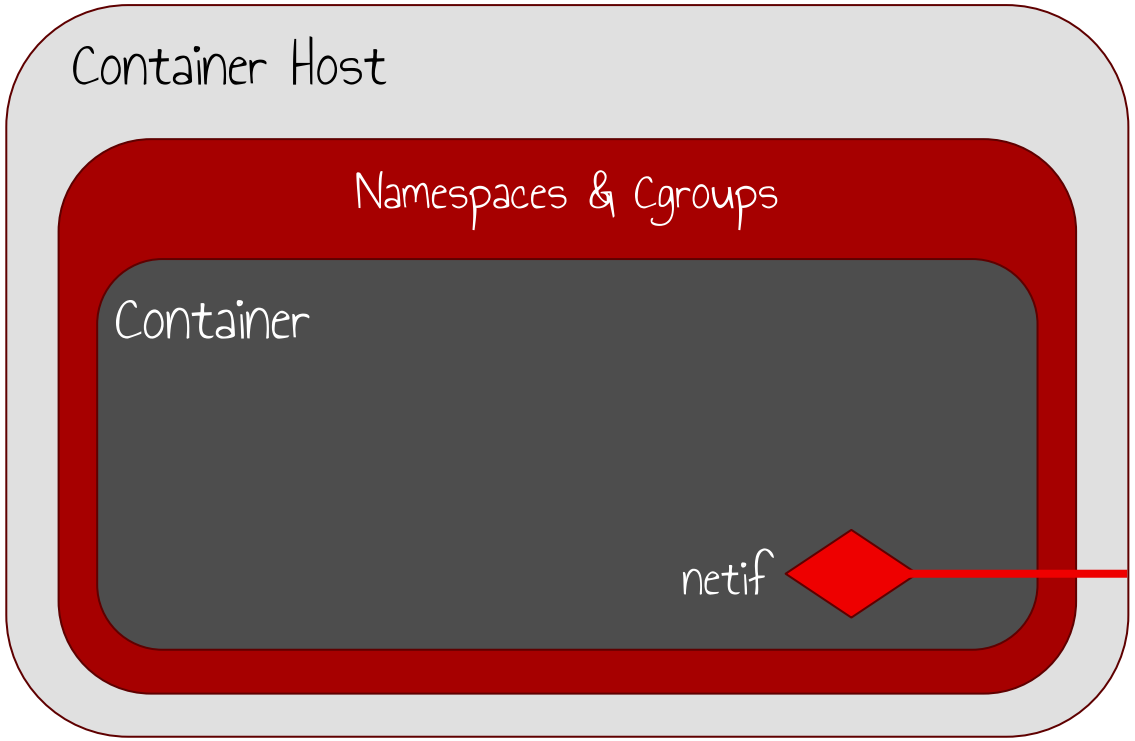
See also:
https://www.youtube.com/watch?v=QMUEtEt1i3I

Dramatis Personae

Containers, virtual machines and networks

Containers

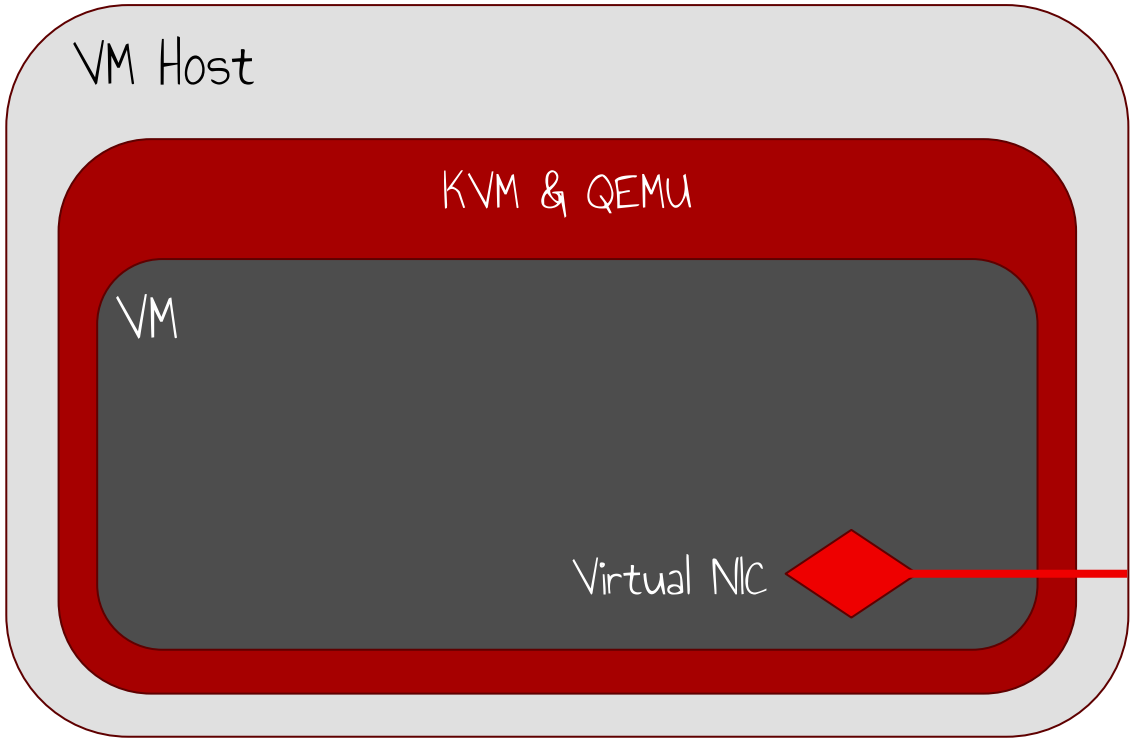
Isolation with namespaces



- Guest Environment
  - Linux userspace
- Isolation Technology
  - Namespaces & Cgroups
- Network
  - Linux network interface
    - Usually veth

Virtual Machines

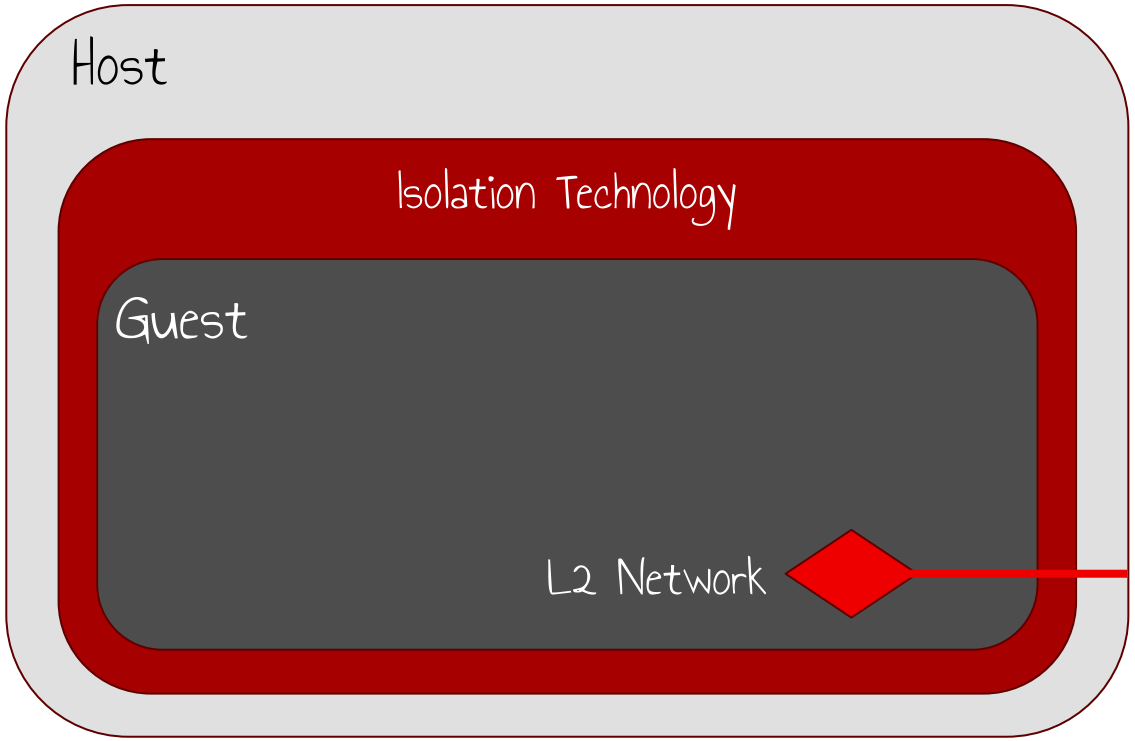
Isolation with KVM



- Guest Environment
  - Virtual hardware
- Isolation Technology
  - KVM + hardware emulation (QEMU)
- Network
  - Emulated or virtual NIC
    - Usually virtio-net

Guests

Isolation in general



- Guest Environment
  - Whatever...
- Isolation Technology
  - Guest can't hurt the host
- Network
  - Operates at L2 (Ethernet) level

Guests and Privilege

Who can make a guest?

Container Isolation

Linux kernel namespaces

- User Namespaces
  - Introduced in Linux 3.8
    - More or less complete by 3.12
  - Anyone can create
    - ... and gain all capabilities within the namespace
- Other namespaces can be created within user namespace
- DEMO

See also:
`man user_namespaces(7)`

Virtual Machine Isolation

KVM and QEMU

- Using KVM isolation requires write access to `/dev/kvm`
  - World writable on most distros
    - Group restricted on others (e.g. Debian)
- QEMU is just a regular process
  - Only needs permissions to pass through host resources
- DEMO

Guest Isolation

General principles

- The Guest can't hurt the Host
  - That's the whole point of isolation
  - So it's safe for anyone to make a guest
- Isolation technologies generally don't require privilege
  - If they can be used unprivileged, they should
    - ...right?

Privilege in Practice

Docker

```
UID        PID    PPID  C STIME TTY          TIME CMD
…
root 1220326       1  0 15:20 ?        00:00:00 /usr/bin/dockerd --host=fd:// --exe
root 1220346 1220326  0 15:20 ?        00:00:00 containerd --config /var/run/docker
dwg  1220534 1220041  0 15:20 pts/3    00:00:00 docker run -it busybox
root 1220603       1  0 15:20 ?        00:00:00 /usr/bin/containerd-shim-runc-v2 -n
…
```

Privilege in Practice

Kubernetes (k3s)

```
UID          PID    PPID  C STIME TTY          TIME CMD
…
root         741       1  2 22:18 ?        00:00:29 /usr/local/bin/k3s agent
root         946     741  1 22:18 ?        00:00:14 containerd
root        2267       1  0 22:18 ?        00:00:00 /var/…/containerd-shim-runc-v2 -nam
root        5392       1  0 22:29 ?        00:00:00 /var/…/containerd-shim-runc-v2 -nam
root        6378       1  0 22:34 ?        00:00:00 /var/…/containerd-shim-runc-v2 -nam
root        6395       1  0 22:34 ?        00:00:00 /var/…/containerd-shim-runc-v2 -nam
root        8002       1  0 22:37 ?        00:00:00 /var/…/containerd-shim-runc-v2 -nam
…
```

Privilege in Practice

Kubernetes (OpenShift)

```
UID          PID    PPID  C STIME TTY          TIME CMD
…
root        2126       1 12 03:43 ?        00:04:50 /usr/bin/crio
root        2178       1  5 03:44 ?        00:01:55 /usr/bin/kubelet --config=/etc/kube
root       36305       1  0 04:17 ?        00:00:00 /usr/bin/conmon … /usr/bin/runc --r
…
```

Privilege in Practice

OpenStack

```
UID        PID    PPID  C STIME TTY          TIME CMD
…
root     44653       1  0  2021 ?        04:00:51 /usr/bin/containerd-shim-runc-v2 -n
root     44673   44653  0  2021 ?        00:00:00 dumb-init --single-child -- kolla_s
root     44687   44673  0  2021 ?      5-21:37:51 /usr/sbin/libvirtd --listen
42436 1210542 1210529  2  2023 ?      7-04:22:23 …/python3 /usr/bin/nova-compute
42436 1710396   44653 47 22:07 ?        00:22:05 /usr/libexec/qemu-kvm -name guest=i
…
```

Why root?

Principle versus practice

- Just plain sloppiness?
- An accident of history?
  - Docker predates user namespaces
  - The rest copied its design
    - Including the specifications
- A crucial feature that needs privilege
  - Networking

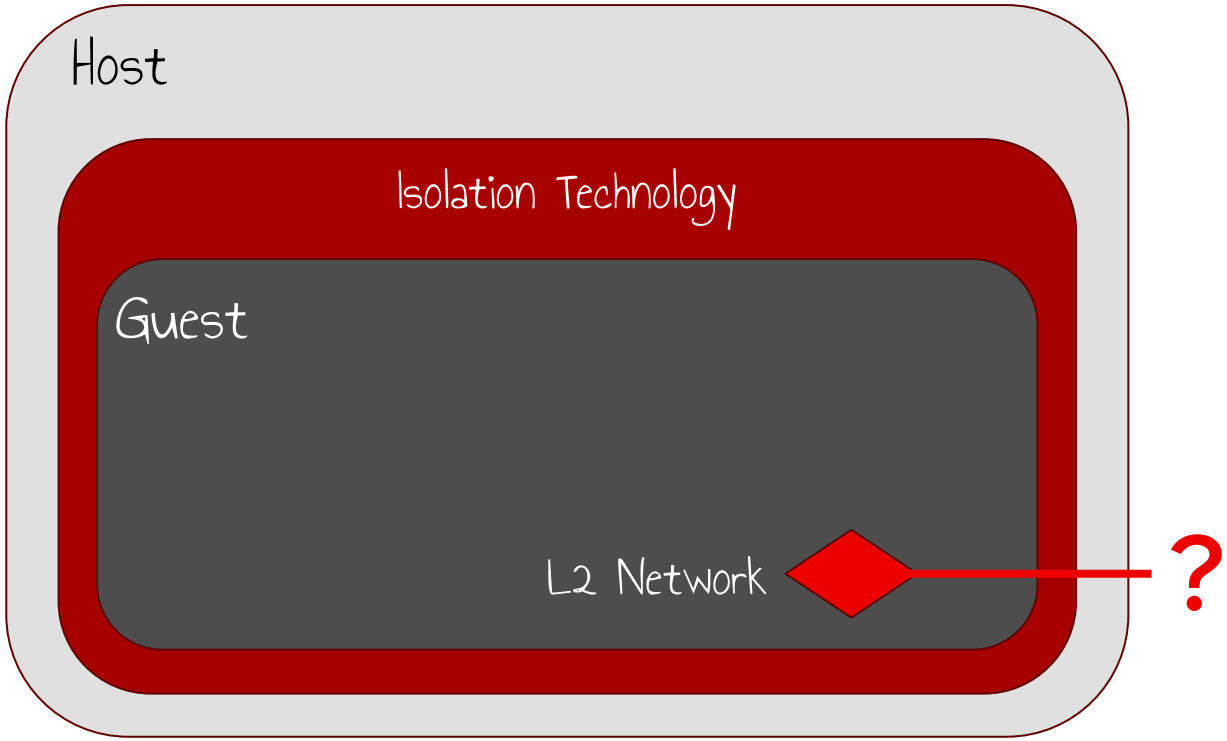
See also:
https://www.youtube.com/watch?v=qXG_cChQgUg
https://rootlesscontaine.rs/

Guest Networking

What connects inside to outside?

Guest Networking

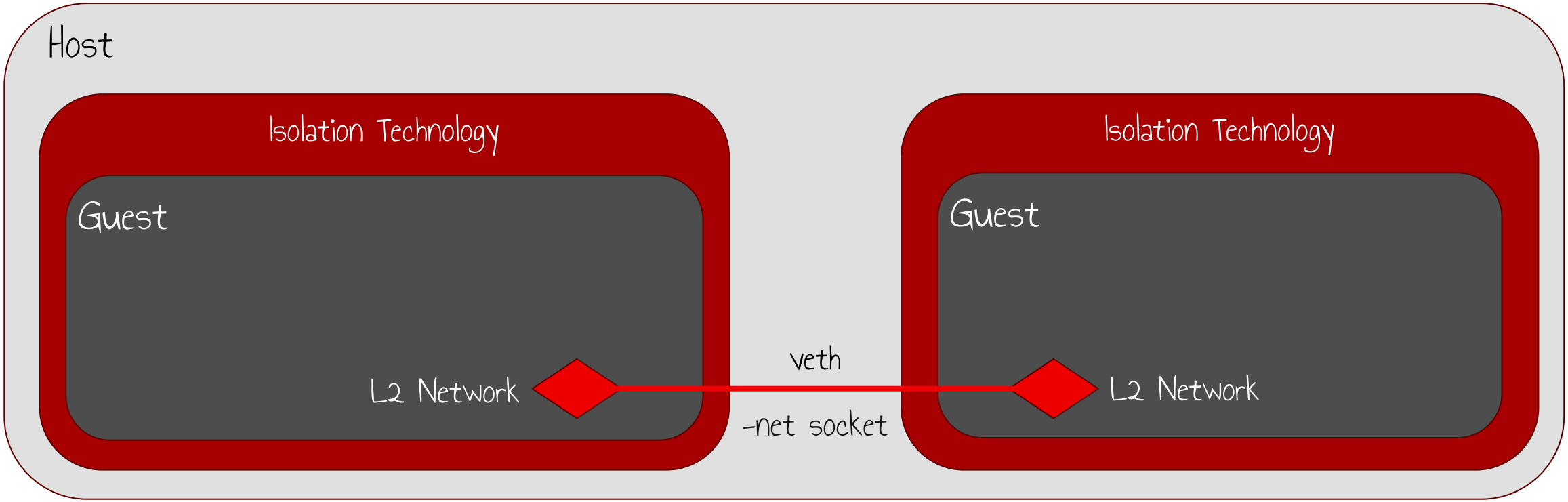
The basic idea



- Most guests need a network connection
  - Possibly to a local cluster
  - Usually to the global internet
- Expect it to look like an L2 link
- How is it “wired up”?

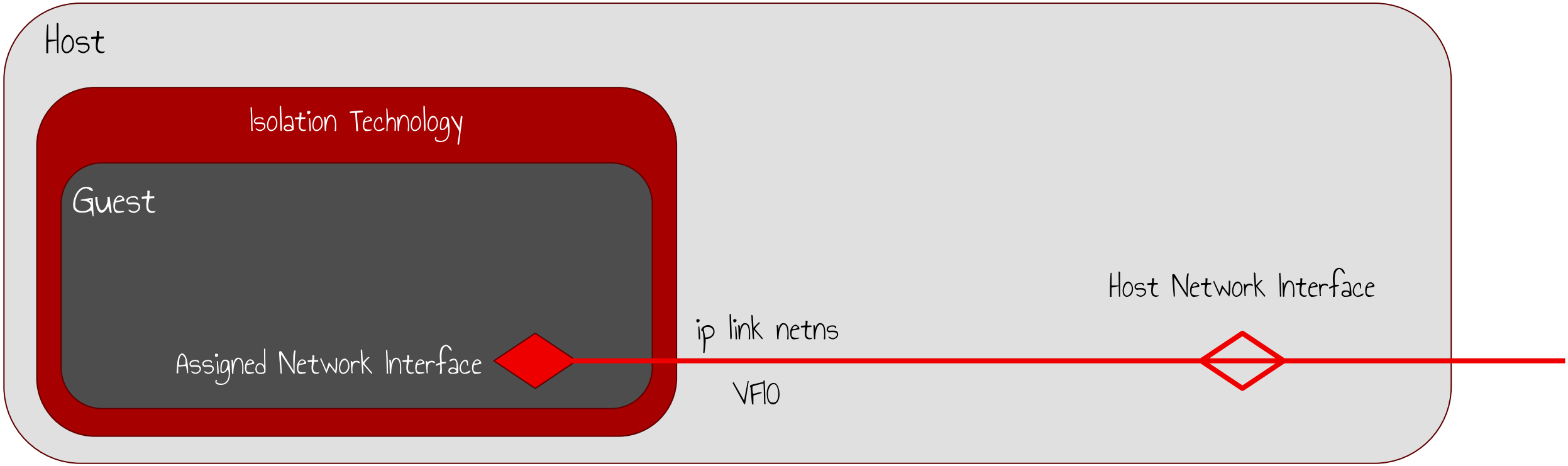
Isolated Networking

No privilege required, but limited usefulness



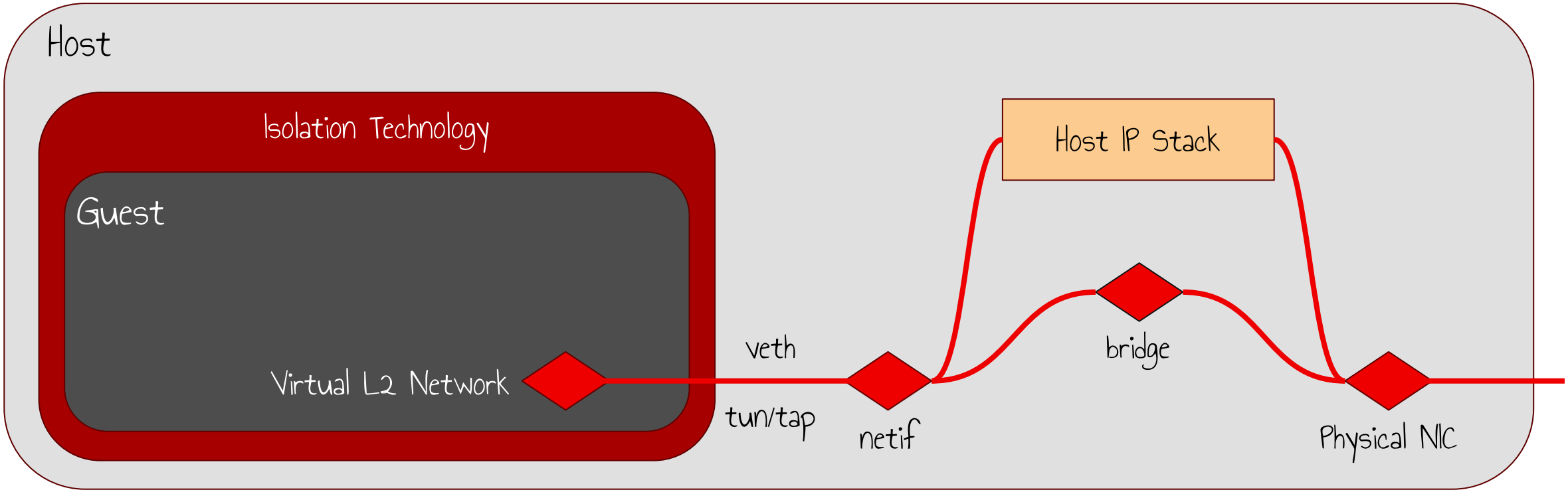
Device or Interface Assignment

Some permissions required, and awkward



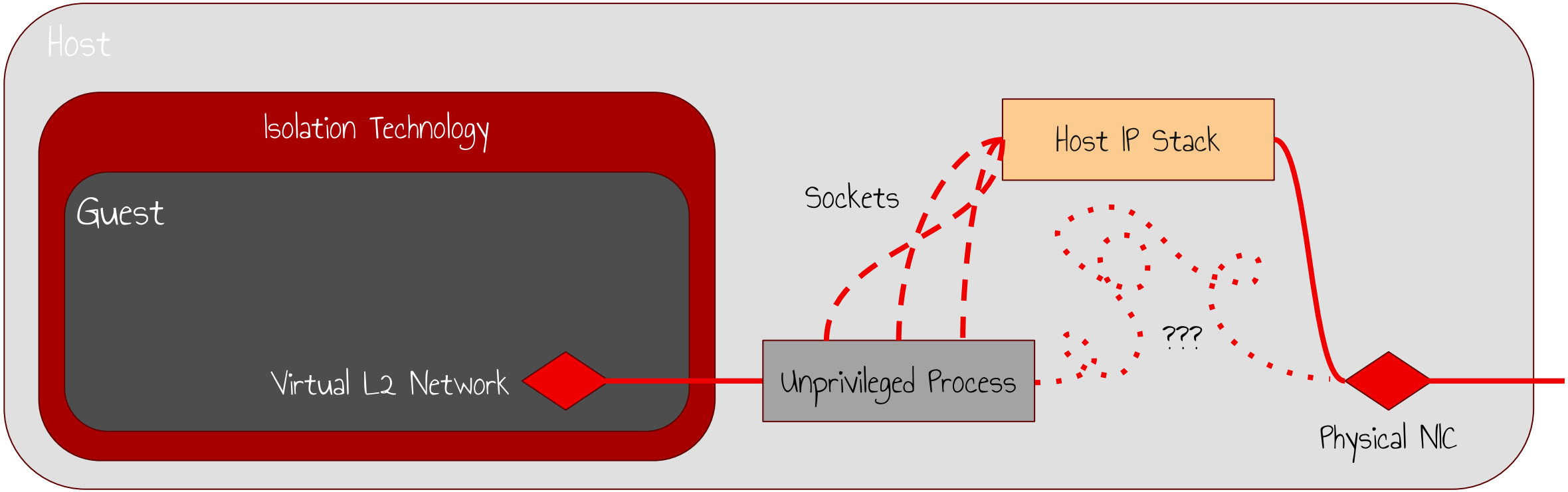
Connection via Host

Convenient and flexible, but needs CAP_NET_ADMIN



Rootless Network

External connectivity without privilege



Slirp

Exploring the past to create a better future?

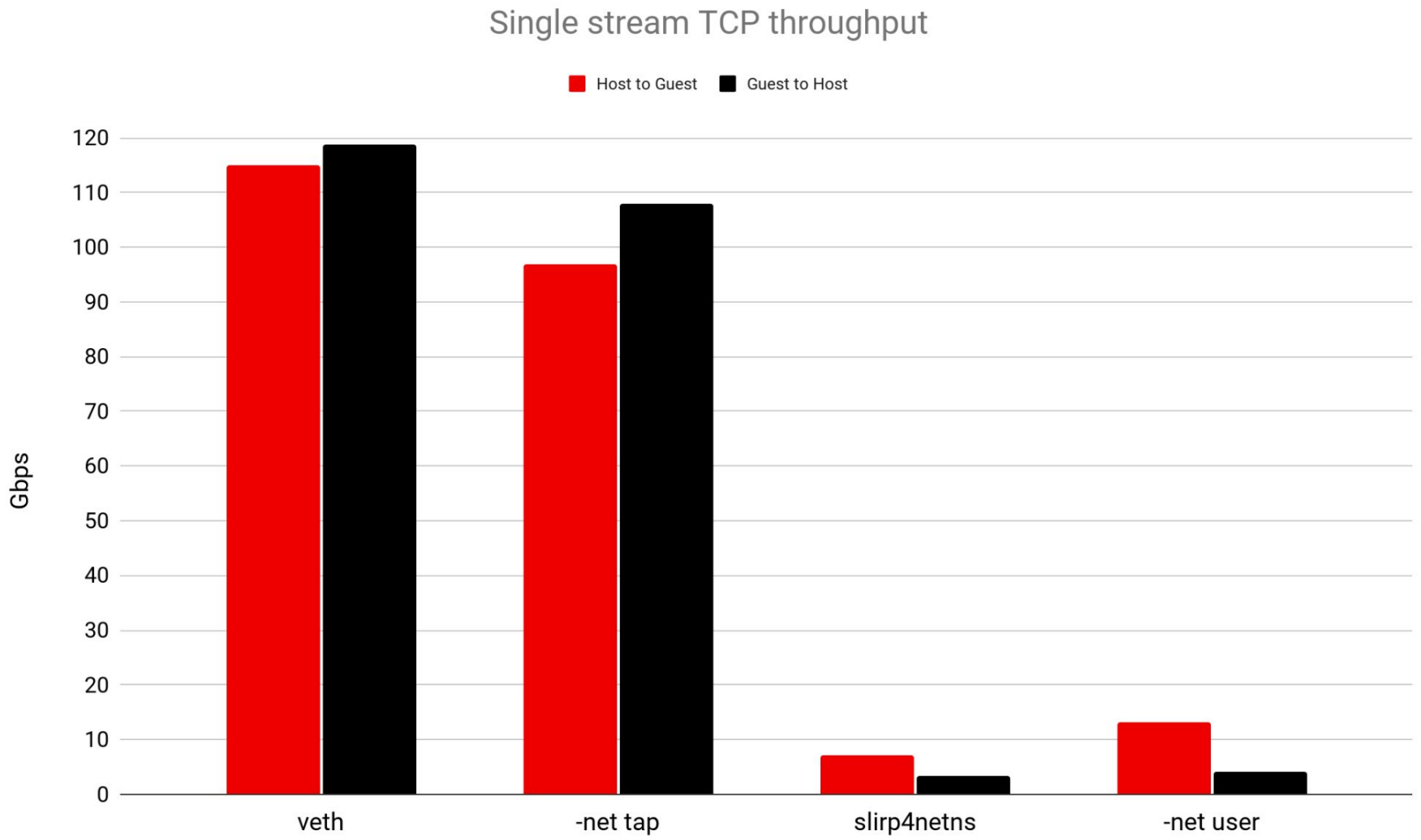
- An L2 ↔ L4 bridge turns out to be a very old idea
  - Used in the ‘90s to do SLIP over dial-up shell accounts
- Resurrected to allow rootless VM networks
  - QEMU `-net user` (based on libslirp)
- And again for rootless container networks
  - `slirp4netns`
  - RootlessKit (`slirp4netns` plus other bits)

See also:
https://www.youtube.com/watch?v=QMUEtEt1i3I

Slirp

Turns out 30 year old technology has some drawbacks

- Everything is NATted
  - It's the obvious choice (no way to allocate addresses)
  - Not great for clusters like Kubernetes
    - Especially service meshes
- Poor security history
  - Particularly of resource leaks
- Limited IPv6 support



`iperf3 -P 1 -l 1M`
ThinkPad P1 Gen5 laptop (10 core/20 thread i7-12800H)
`net.core.rmem_max` and `net.core.wmem_max` set to 16MiB

Red Hat

The Status Quo

Rootless or fast, pick one

Rootless Networking

- "Quick and dirty"
  - Great for experimentation
  - Useful during development
- For production
  - Usually too slow

Rootful Networking

- Requires system-wide configuration
  - Awkward for experimentation
- For production
  - Increases attack surface
  - But the only practical choice

passt & pasta

Newer, faster, better

passt & pasta

One binary, two modes

pasta

- Userspace, unprivileged networking
  - for **containers and namespaces**
- Replacement for `slirp4netns`
  - With or without RootlessKit
- Connects to namespace
  - via tap device
  - maybe other options in future

passt

- Userspace, unprivileged networking
  - for **QEMU Virtual Machines**
- Replacement for `qemu -net user`
- Connects to QEMU
  - via Unix socket
  - via vhost-user (work in progress)

passt & pasta

A modern L2 ↔ L4 bridge

Trivia

- "Plug a Simple Socket Transport"
  - (and wordplay in German)
- Originally written by Stefano Brivio
  - Starting late 2020
- I joined project May 2022
- Significant contributions from
  - Jon Maloy, Laurent Vivier
- Written in C

Design Goals

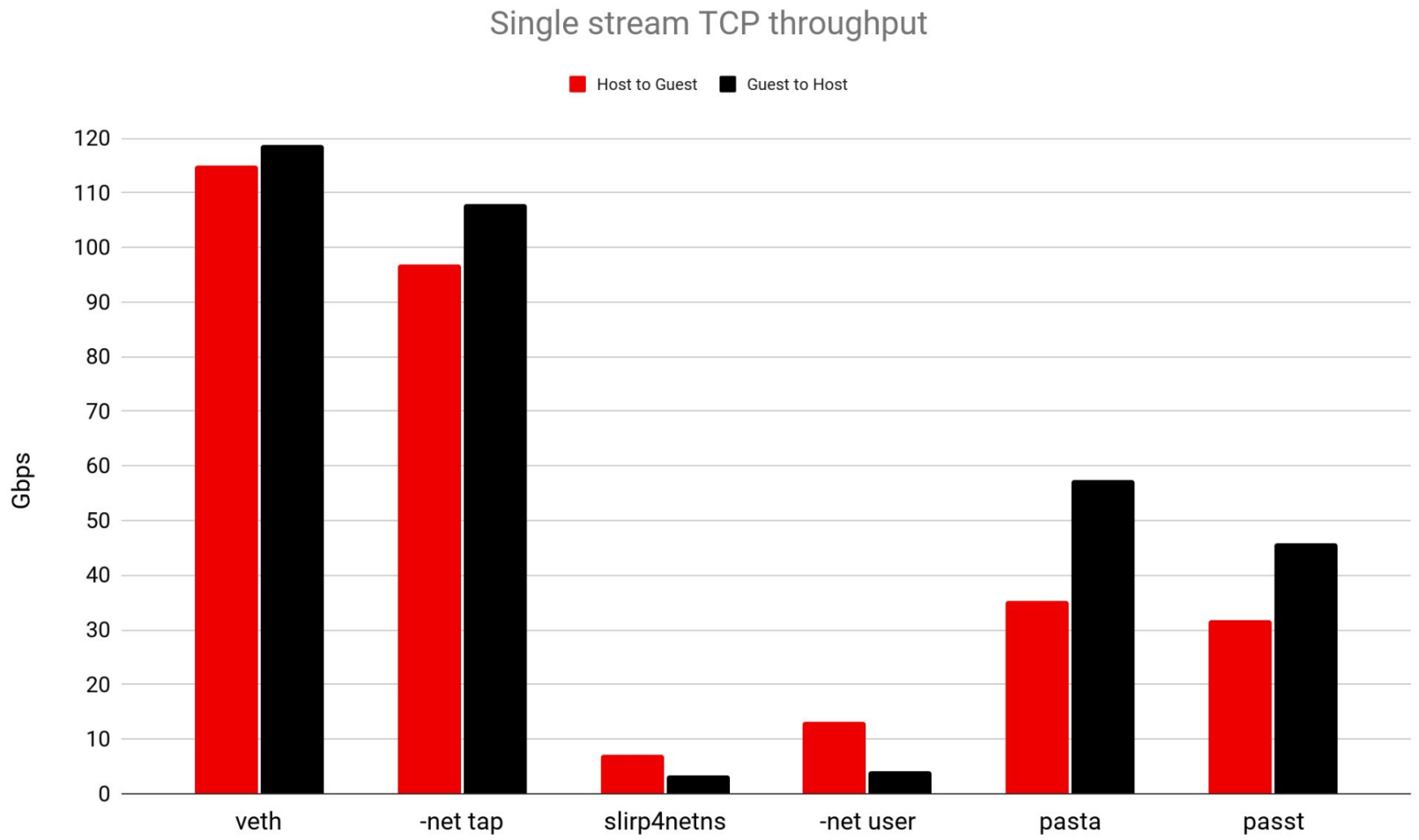
- No dependencies (except libc & kernel)
- No NAT
  - Well… minimal NAT
- No dynamic memory allocation
- IPv6 as first class citizen
- Security conscious
- *Reasonably performant*

See also:
https://passt.top

Performance

Techniques to improve

- TCP
  - ~64kiB MTU advertised to guest
    - Segments coalesced and batched as needed
  - Supports window scaling
    - Slirp only allows ~64kiB “in flight”
- UDP
  - `sendmmsg()` / `recvmmsg()` to batch syscalls
- AVX2 checksum routines (on x86_64)
  - Could be implemented for other SIMD platforms



`iperf3 -P 1 -l 1M`
ThinkPad P1 Gen5 laptop (10 core/20 thread i7-12800H)
`net.core.rmem_max` and `net.core.wmem_max` set to 16MiB

Red Hat

Potential improvements

Cache tuning

- Recently tested some changes which altered batching of packets
  - Feared it would greatly slow down throughput
  - … but instead substantially sped things up
- Appears to be due to better cache behaviour
  - Looks like there's some significant scope for tuning
  - Maybe as much as 50% for host to guest traffic
- Still figuring out what's going on
  - … so it's hard to be sure of much

Potential improvements

SO_PEEK_OFF

- passt/pasta uses MSG_PEEK heavily on host to guest path
  - Keeps data in socket buffer until acknowledged
  - Avoids allocating per-stream buffers
- Have to re-read all buffered data on each peek
- SO_PEEK_OFF
  - Kernel feature that exists on Unix sockets
  - Jon Maloy implemented for TCP sockets
- Avoids extra kernel to user copies
  - Perhaps 20% improvement for host to guest

See also:
https://git.kernel.org/pub/scm/linux/kernel/git/netdev/net-next.git/commit/?id=05ea491641d338422edbd87af3b311c16ea762f5

Potential improvements

vhost-user

- Unix domain socket to qemu is a bottleneck
  - Memory copy from passt to kernel, then kernel to qemu
- vhost-user is a different network backend
  - passt shares memory with VM guest
  - passt can act like a high-performance NIC's descriptor ring
- Laurent Vivier has draft patches
- Also allows us to skip checksums
  - Significant portion of our CPU usage

Potential Improvements

Long term

VDUSE

- Replacement for tuntap device
- Based on vDPA
- Similar mechanics to vhost-user
  - Bring its benefits to pasta
- Relatively easy once vhost-user is done
  - … we hope

Multithreading

- Currently single threaded
- Potentially large benefits
- Requires significant work
  - Not going to happen for a while

See also:
https://www.redhat.com/en/blog/introducing-vduse-software-defined-datapath-virtio

Beyond Single Stream Throughput

Other aspects of performance

- Multiple TCP streams
  - Veth gets faster until ~4 streams
  - Others stay pretty static up until ~8 streams
- Latency
  - Passt & slirp are similar, around twice veth
- UDP
  - Tricky to measure
    - Tools surprisingly thin on the ground
  - Haven’t worked on it much yet

Alternatives

Rootless networking without an L2 ↔ L4 bridge

Privileged helpers

- Configure netifs on behalf of rootless
  - Libvirt does this now for VMs
- Pro:
  - Simple, good performance
- Con:
  - Requires system-wide configuration
  - Guest still controls a netif
    - … so it's not entirely unprivileged

Socket Replacement

- `bypass4netns`
- `libkrun` Transparent Socket Impersonation
- Pro:
  - Good performance
- Con:
  - Weakens isolation
  - Only allows guest userspace traffic
    - No guest routing, or nftables

See also:
https://github.com/rootless-containers/bypass4netns

Demos

How do I use the new toy?

Installation

- Packages exist for many distros
  - Alpine, Arch, CentOS Stream, Debian, Fedora, Gentoo, GNU Guix, NixOS, OpenSUSE, Ubuntu, Void Linux
  - EPEL, Mageia (unofficial)
- Also easy and fast to build locally
  - No dependencies beyond libc
- DEMO

pasta standalone

- Instantly create a netns with external connectivity
- Useful for quick experimentation
- DEMO

podman

- Podman has supported pasta since 4.4
- Default rootless network option from podman 5.0
  - Released March 28 2024
- DEMO

See also:
https://www.redhat.com/en/blog/podman-50-unveiled

QEMU

- Easy with Qemu 7.2.0 or later
  - Added `-net stream` / `-net dgram`
    - Allows Unix socket for network backend
  - Possible before, but requires passing fds around
- Hope to extend qemu with `-net passt` or similar
- DEMO

libvirt

- Support added in libvirt 9.00
  - Some SELinux bugs, fixed by 9.2.0
- DEMO

Conclusion

Summary

- You used to face a choice: unprivileged or impractical
- passt & pasta make rootless networking practical for more cases
  - Allows more real world uses cases
  - Allows less differences between experimental and “real”
- You can try it now!
  - Widely available
  - Integrated with podman, libvirt amongst others

Credits and Contributions

... and questions!

Thanks to:

- Stefano Brivio
  - Created the project
  - Reviewed these slides
- Laurent Vivier
  - Working on vhost-user
- Jon Maloy
  - Implemented TCP SO_PEEK_OFF
  - ... and fixed some nasty TCP bugs

- Website:
  - https://passt.top
- Bugzilla
  - https://bugs.passt.top
- IRC
  - `#passt` on Libera.Chat
- Mailing lists:
  - `passt-user@passt.top`
  - `passt-dev@passt.top`

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat