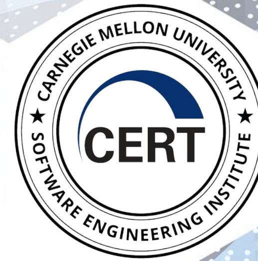


Virtual Networking with Linux

Gabriel L. Somlo, Ph.D.

CERT / SEI
Carnegie Mellon University
Pittsburgh, PA 15213

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Copyright 2022 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

CERT® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM22-0863

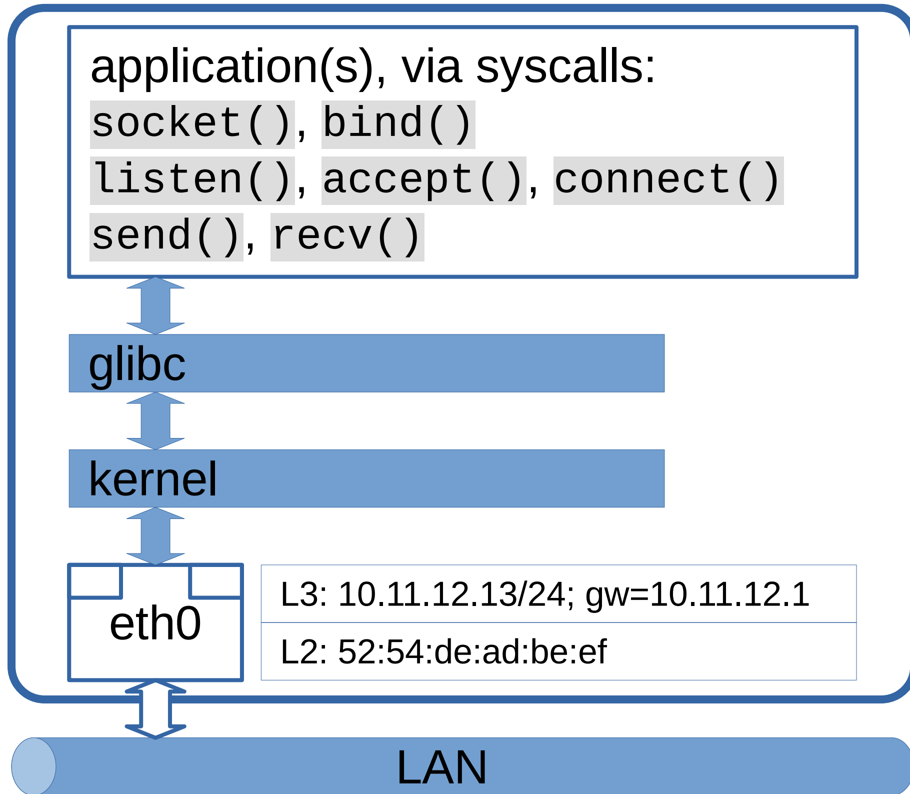
Motivation

- Understand “virtual host” net access options:
  - Containers
  - Virtual machines
  - Simulators
  - custom-coded applications

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Typical Host Network Interaction



- Programs running on host use standard network API
- Host may have multiple interfaces, optionally forwarding traffic between them

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Virtual Network "cables": *veth*

```
ip link add ve_a [netns ns-a] \
    type veth \
    peer name ve_b [netns ns-b]
```

- Any data received on `ve_a` immediately transmitted on `ve_b`
  - and vice versa
- Network namespaces (`netns`) useful to connect containers!

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Virtual Network “switches”: *bridge*

- Software-emulated Layer-2 (Ethernet) switch

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Virtual Network “switches”: *bridge*

```
ip link add br0 type bridge
```

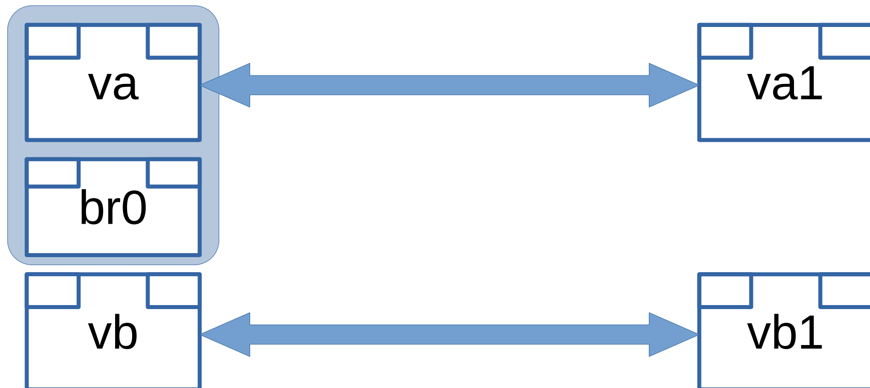
- Software-emulated Layer-2 (Ethernet) switch
- Setup could use `brctl [addbr | addif]` as alternative to `ip link`
- Use `brctl show` to display information

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Virtual Network “switches”: *bridge*

```
ip link add br0 type bridge

ip link set va master br0
```



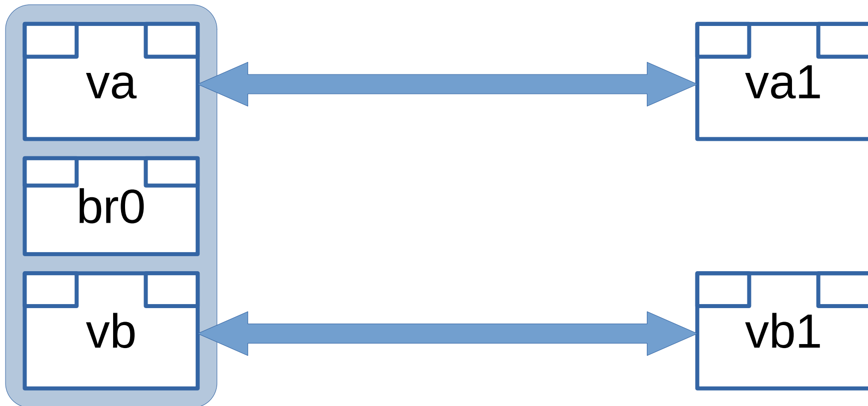
- Software-emulated Layer-2 (Ethernet) switch
- Setup could use `brctl [addbr | addif]` as alternative to `ip link`
- Use `brctl show` to display information

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Virtual Network “switches”: *bridge*

```
ip link add br0 type bridge

ip link set va master br0
ip link set vb master br0

for i in br0 va vb va1 vb1; do
    ip link set up $i
done
```

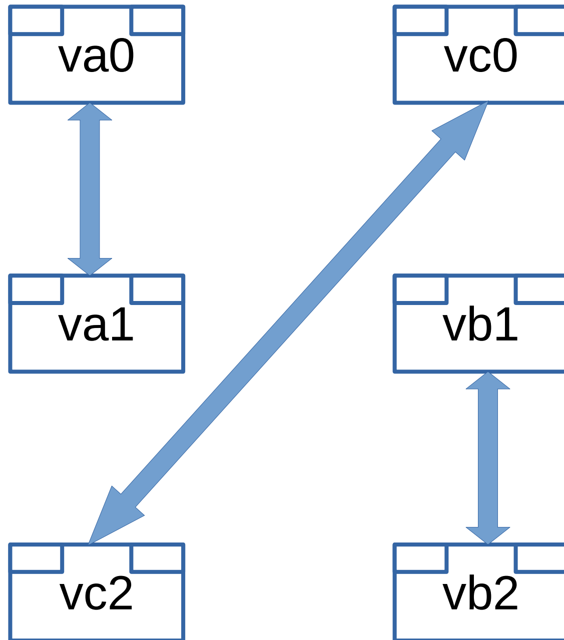


- Software-emulated Layer-2 (Ethernet) switch
- Setup could use `brctl [addbr | addif]` as alternative to `ip link`
- Use `brctl show` to display information

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Example: Studying STP Behavior

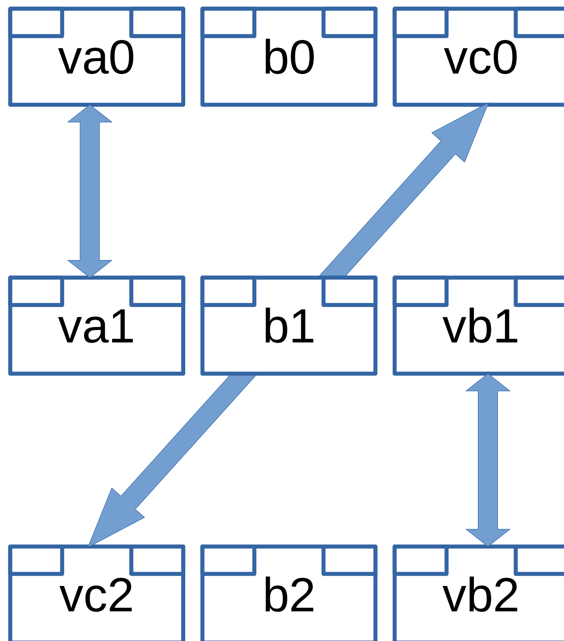



```
ip link add va0 type veth peer name va1
ip link add vb1 type veth peer name vb2
ip link add vc2 type veth peer name vc0
```


© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Example: Studying STP Behavior

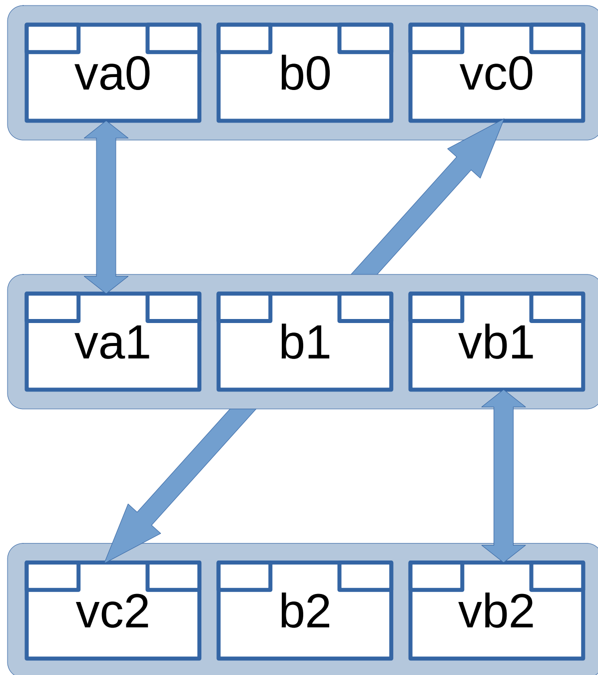



```
ip link add va0 type veth peer name va1
ip link add vb1 type veth peer name vb2
ip link add vc2 type veth peer name vc0

for i in 0 1 2; do
 ip link add b$i type bridge; brctl stp b$i on
done
```


© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved
for public release and unlimited distribution.
DM22-0863

Example: Studying STP Behavior

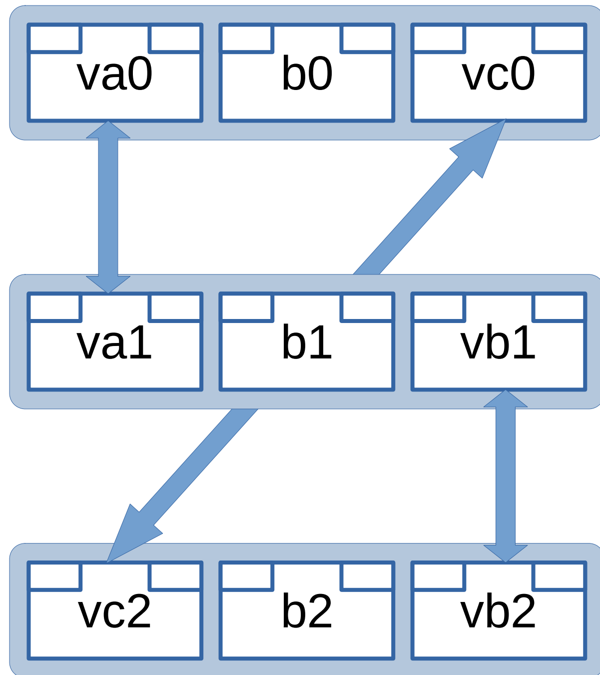



```
ip link add va0 type veth peer name va1
ip link add vb1 type veth peer name vb2
ip link add vc2 type veth peer name vc0

for i in 0 1 2; do
 ip link add b$i type bridge; brctl stp b$i on
done

ip link set va0 master b0
…
ip link set vc2 master b2
```


© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved
for public release and unlimited distribution.
DM22-0863

Example: Studying STP Behavior




```
ip link add va0 type veth peer name va1
ip link add vb1 type veth peer name vb2
ip link add vc2 type veth peer name vc0

for i in 0 1 2; do
 ip link add b$i type bridge; brctl stp b$i on
done

ip link set va0 master b0
…
ip link set vc2 master b2

for i in b0 b1 b2 va0 va1 vb1 vb2 vc2 vc0; do
 ip link set up $i
done

tshark -i va0
```


© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

TUN/TAP Linux Driver

- Programs can bypass the host network API
  - Request their own IP (TUN) or Ethernet (TAP) network interface…

    … which may be routed (or bridged) to existing interfaces
- Used by, e.g., ssh[d], vtun, gns3, CORE (greybox), and many, many more
- Details in kernel Documentation/networking/tuntap.rst

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Example: Request a TAP Interface

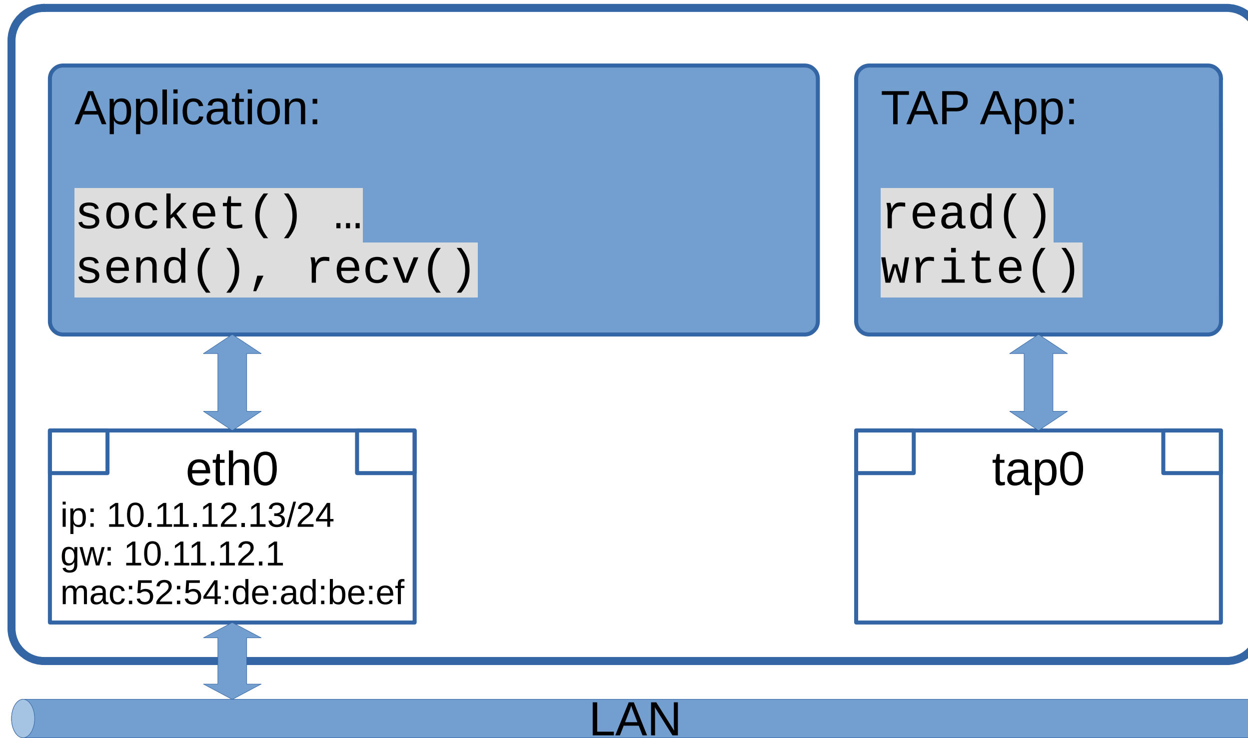
```
int tap_alloc(char *dev) {
  struct ifreq ifr;
  int fd, err;

  if ((fd = open("/dev/net/tun", O_RDWR)) < 0) return fd;

  memset(&ifr, 0, sizeof(ifr));
  ifr.ifr_flags = IFF_TAP;

  if (*dev) strscpy_pad(ifr.ifr_name, dev, IFNAMSIZ);

  if ((err = ioctl(fd, TUNSETIFF, (void *)&ifr)) < 0) {
    close(fd);
    return err;
  }

  strcpy(dev, ifr.ifr_name);
  return fd;
}
```

```
int main(…) {
  …

  int fd;

  …

  fd = tap_alloc("tap0");

  …

  read(fd, …);
  …
  write(fd, …);

  …

}
```


© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Bridge TAP App to Host Network

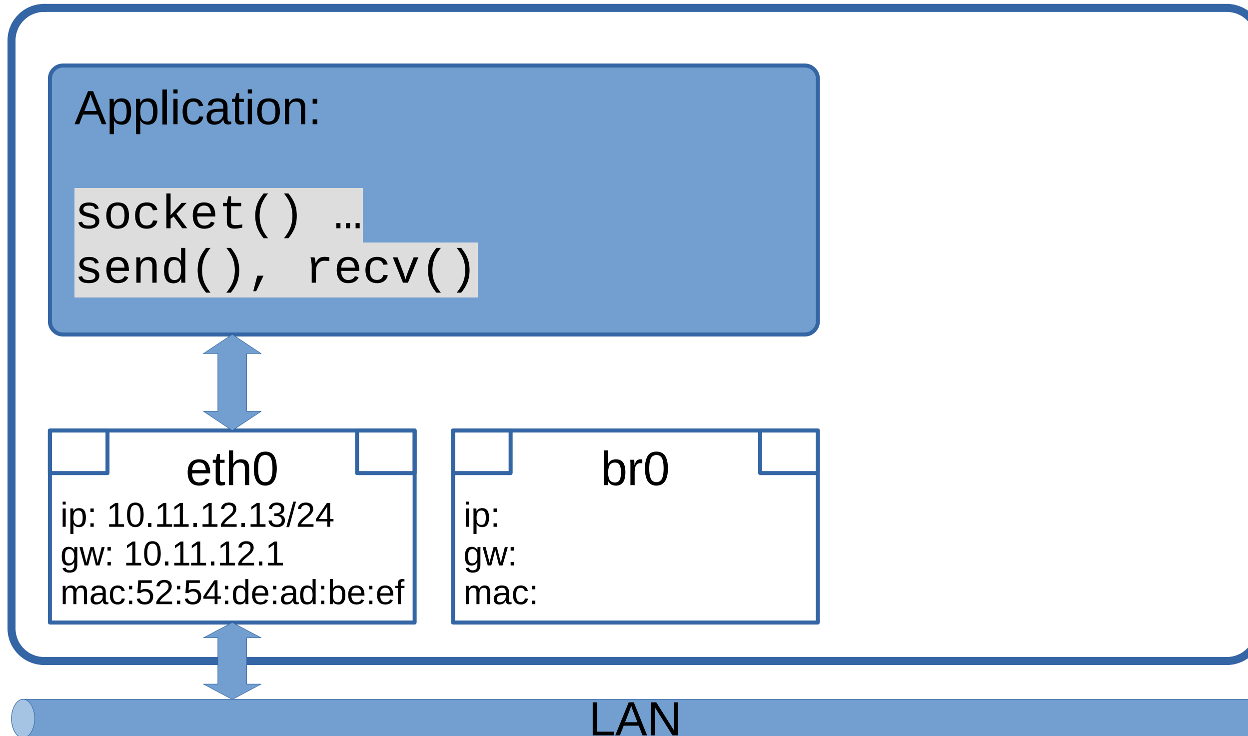


- TAP app may also be using “regular” host network API
  - e.g., VPN endpoint (client, server)

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Adding a Bridge to Host Netwrk API

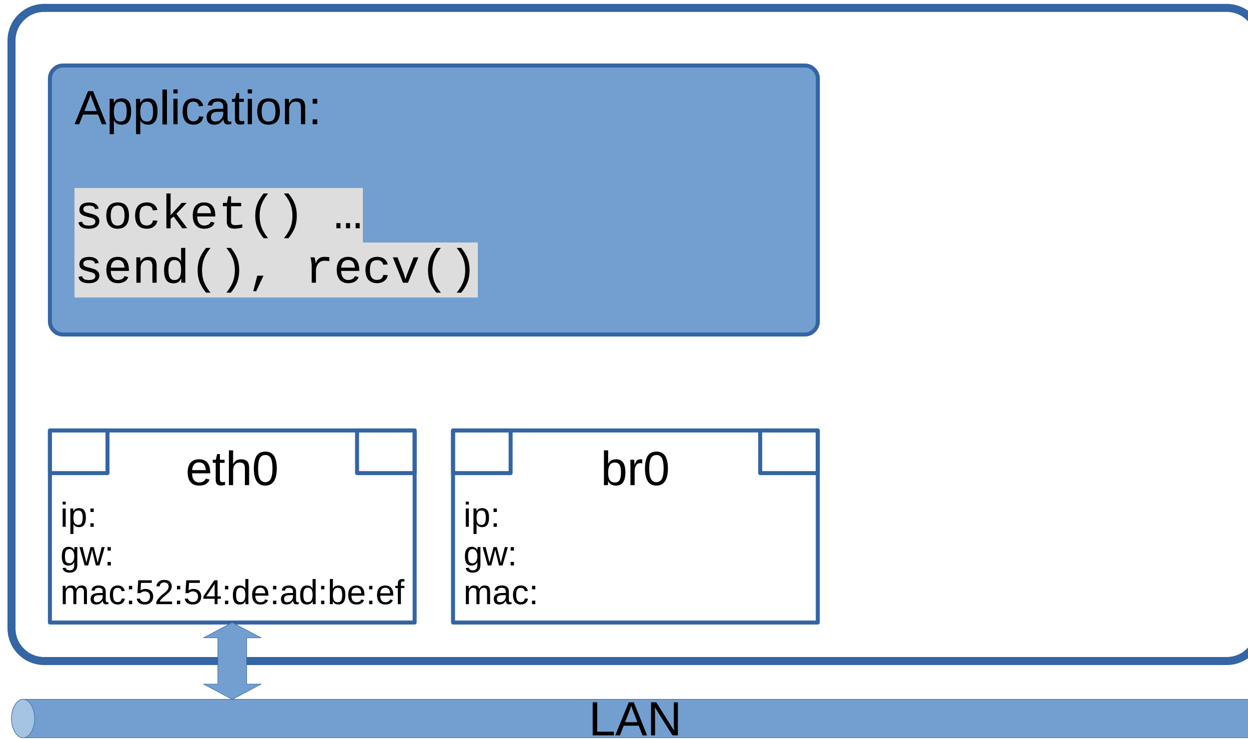



```
ip link add br0 type bridge
ip link set up br0
```


© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved
for public release and unlimited distribution.
DM22-0863

Clear *eth0* Layer-3 (IP) Settings

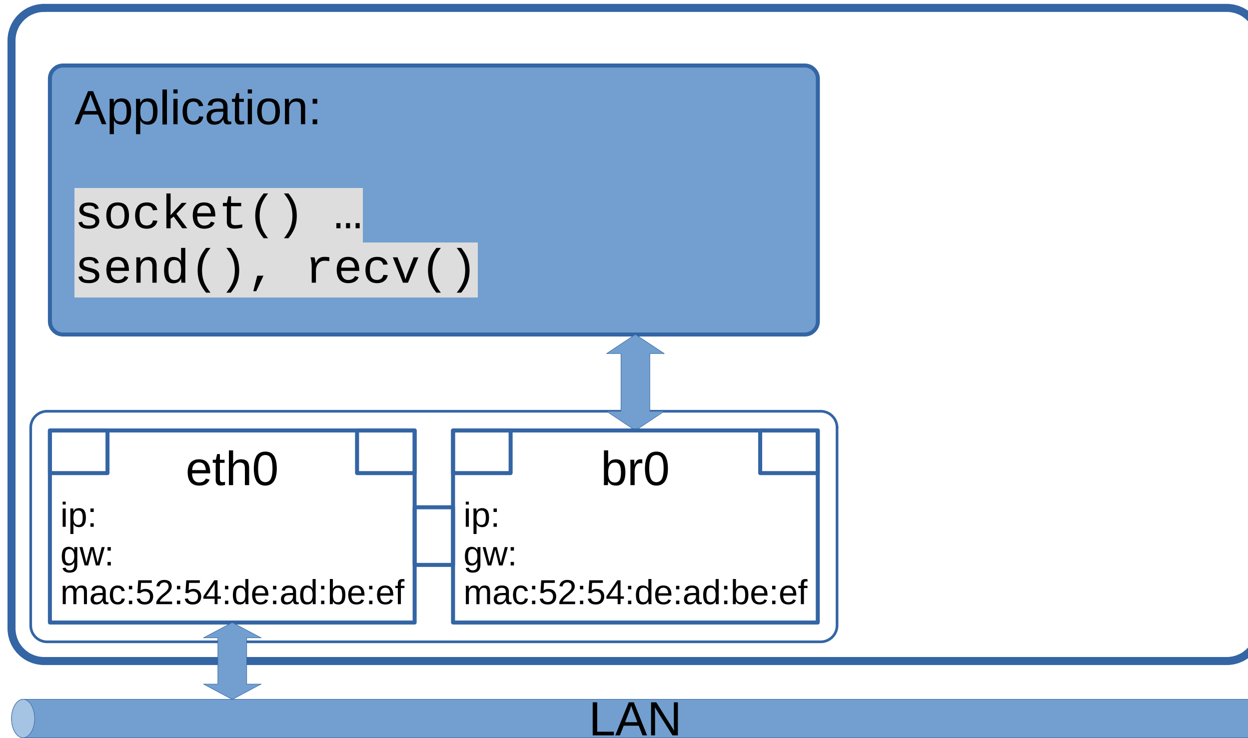



```
ip link add br0 type bridge
ip link set up br0

ip addr flush eth0
```


© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Make *eth0* a Bridge Port of *br0*



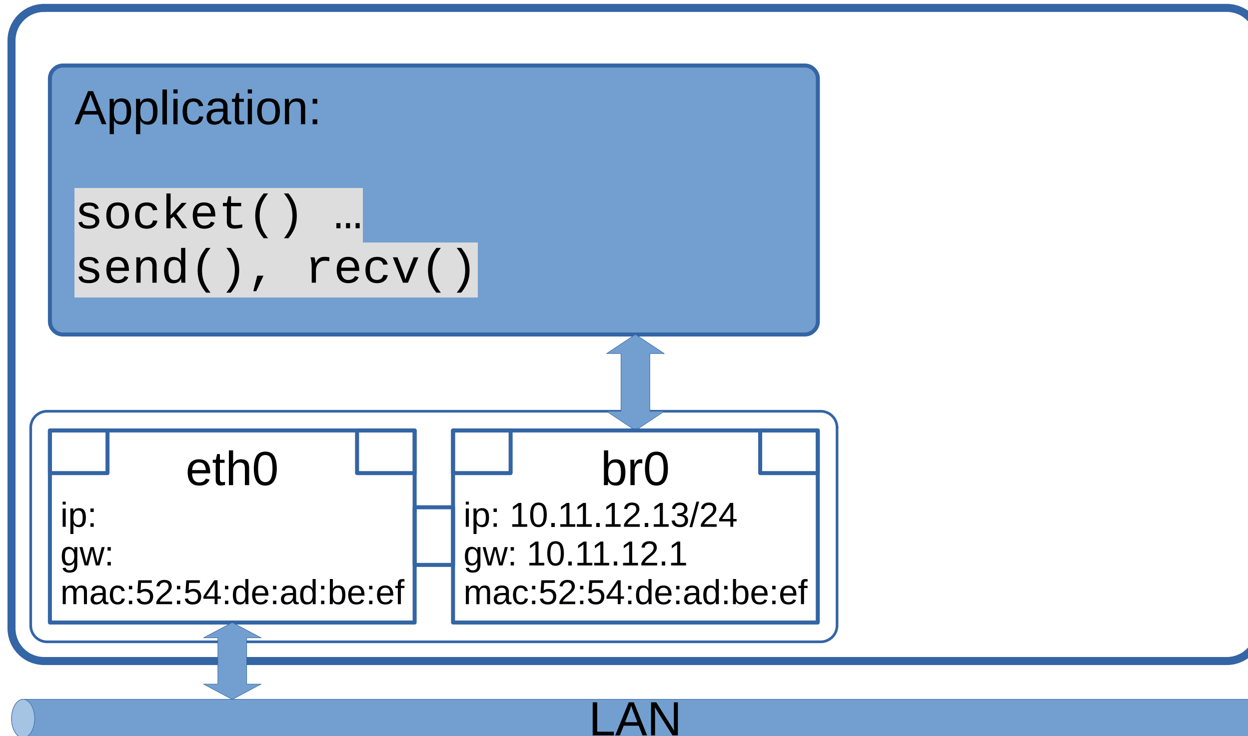

```
ip link add br0 type bridge
ip link set up br0

ip addr flush eth0

ip link set eth0 master br0
```

- By default, a bridge inherits the MAC addr. of its *first* slave port* !

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

Restore Host IP Config, on *br0*



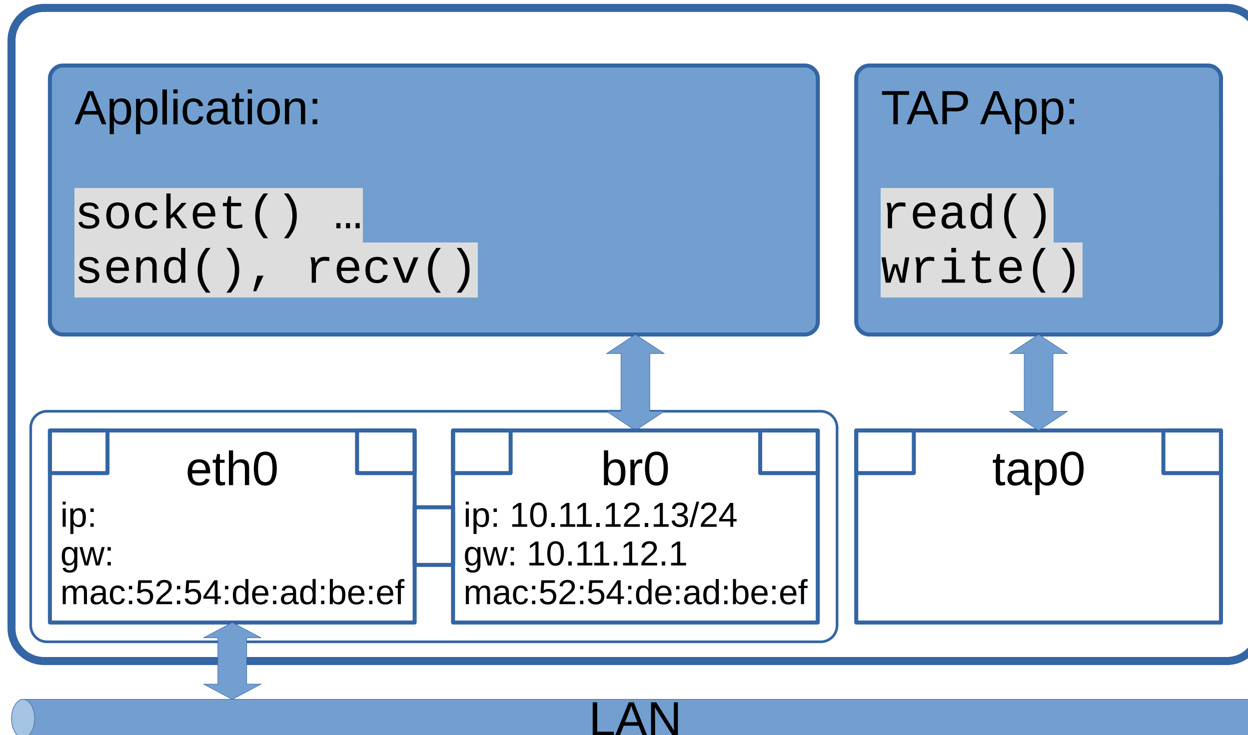

```
ip link add br0 type bridge
ip link set up br0

ip addr flush eth0

ip link set eth0 master br0

dhclient -i br0
```

- Change is transparent to both `socket()` API and to LAN!

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

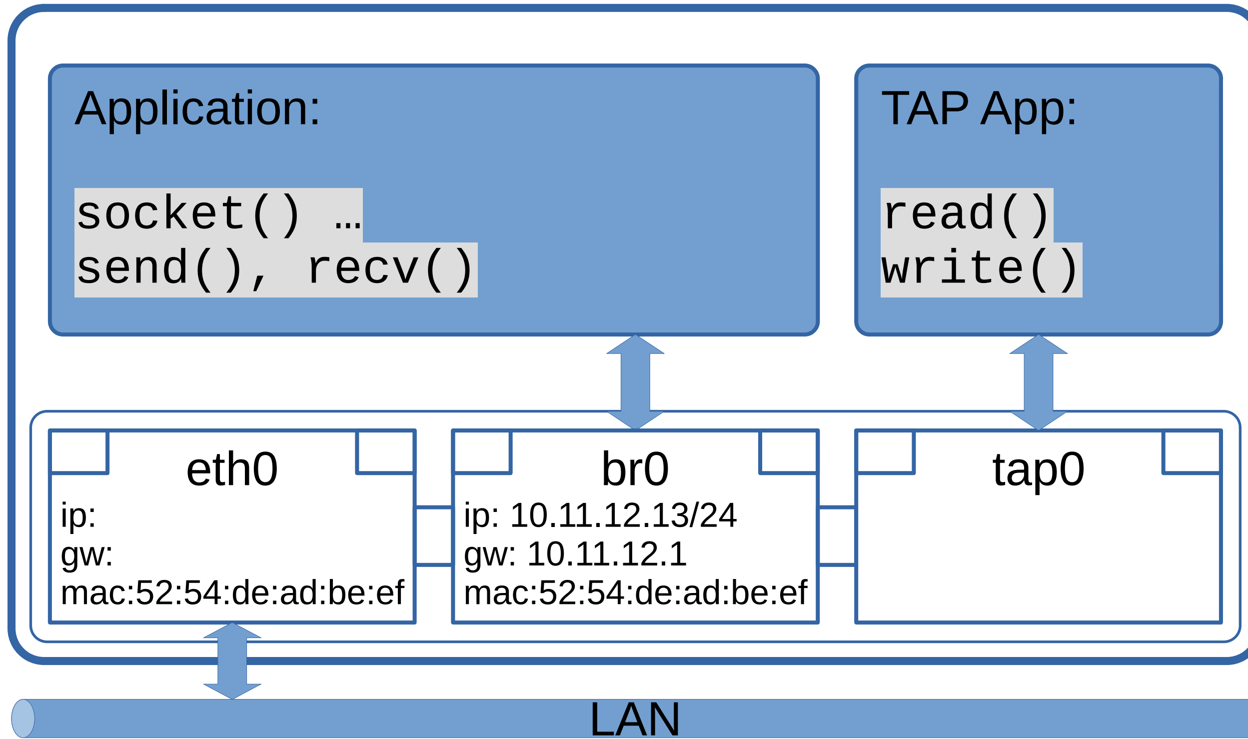
“Patching In” a TAP Application



© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution. DM22-0863

“Patching In” a TAP Application




```
ip link set tap0 master br0
```

- TAP application may likely do this programmatically!

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863

A Word on “Inherit First MAC” Policy

- It’s the *traditional* default behavior
- *systemd* changed behavior to “persistent” auto-generated MAC address for new bridges
  - To restore traditional “inherit-first-mac” behavior, create file `/etc/systemd/network/98-bridge-inherit-mac.link`:

```
[Match]
Type=bridge

[Link]
MACAddressPolicy=none
```

- Attempts in progress to restore (some) sanity:
  - https://fedoraproject.org/wiki/Changes/MAC_Address_Policy_none
  - https://bugzilla.redhat.com/show_bug.cgi?id=1834547

© 2022 Carnegie Mellon University

[DISTRIBUTION STATEMENT A] Approved for public release and unlimited distribution.
DM22-0863