Network bridge

A bridge is a piece of software used to unite two or more network segments. A bridge behaves like a virtual network switch, working transparently (the other machines do not need to know or care about its existence). Any real devices (e.g. `eth0`) and virtual devices (e.g. `tap0`) can be connected to it.

Related articles

Bridge with netctl

Network_configuration#Bo

This article explains how to create a bridge that contains at least an ethernet device. This is useful for things like the bridge mode of **QEMU**, setting a software based access point, etc.

Contents

Creating a bridge

There are a number of ways to create a bridge.

With iproute2

This section describes the management of a network bridge using the *ip* tool from the iproute2 (https://www.archlinux.org/packages/?name=iproute2) package, which is included in the base (https://www.archlinux.org/groups/x86_64/base/) group.

Create a new bridge and change its state to up:

```
# ip link add name bridge_name type bridge
# ip link set bridge_name up
```

To add an interface (e.g. eth0) into the bridge, its state must be up:

```
# ip link set eth0 up
```

Adding the interface into the bridge is done by setting its master to `bridge_name` :

```
# ip link set eth0 master bridge_name
```

To show the existing bridges and associated interfaces, use the *bridge* utility (also part of iproute2 (https://www.archlinux.org/packages/?name=iproute2)). See bridge(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/bridge.8) for details.

```
# bridge link
```

This is how to remove an interface from a bridge:

```
# ip link set eth0 nomaster
```

The interface will still be up, so you may also want to bring it down:

```
# ip link set eth0 down
```

To delete a bridge issue the following command:

```
# ip link delete bridge_name type bridge
```

This will automatically remove all interfaces from the bridge. The slave interfaces will still be up, though, so you may also want to bring them down after.

With bridge-utils

This section describes the management of a network bridge using the legacy *brctl* tool from the **bridge-utils (https://www.archlinux.org/packages/?name=bridge-utils)** package, which is available in the **official repositories**. See **brctl(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/brctl.8)** for full listing of options.

Create a new bridge:

```
# brctl addbr bridge_name
```

Add a device to a bridge, for example `eth0`:

```
# brctl addif bridge_name eth0
```

Note: Adding an interface to a bridge will cause the interface to lose its existing IP address. If you're connected remotely via the interface you intend to add to the bridge, you will lose your connection. This problem can be worked around by scripting the bridge to be created at system startup.

Show current bridges and what interfaces they are connected to:

```
$ brctl show
```

Set the bridge device up:

```
# ip link set up dev bridge_name
```

Delete a bridge, you need to first set it to *down*:

```
# ip link set dev bridge_name down
# brctl delbr bridge_name
```

Note: To enable the **bridge-netfilter (http://ebtables.netfilter.org/documentation/bridge-nf.html)** functionality, you need to manually load the `br_netfilter` module:

```
# modprobe br_netfilter
```

See also **Kernel modules#Automatic module handling**.

With netctl

See **Bridge with netctl**.

With systemd-networkd

See **systemd-networkd#Bridge interface**.

With NetworkManager

Gnome's NetworkManager can create bridges, but currently will not auto-connect to them or slave/attached interfaces. Open Network Settings, add a new interface of type Bridge, add a new bridged connection, and select the MAC address of the device to attach to the bridge.

Now, find the UUID of the attached device (by default named "bridge0 slave 1"):

```
$ nmcli connection
```

Finally, enable that connection:

```
$ nmcli con up <UUID>
```

If NetworkManager's default interface for the device you added to the bridge connects automatically, you may want to disable that by clicking the gear next to it in Network Settings, and unchecking "Connect automatically" under "Identity."

Assigning an IP address

When the bridge is fully set up, it can be assigned an IP address:

```
# ip addr add dev bridge_name 192.168.66.66/24
```

Tips and tricks

Wireless interface on a bridge

To add a wireless interface to a bridge, you first have to assign the wireless interface to an access point or start an access point with **hostapd**. Otherwise the wireless interface will not be added to the bridge.

See also **Bridging with a wireless NIC (https://wiki.debian.org/BridgeNetworkConnections#Bridging_with_a_wireless_NIC)** on Debian wiki.

Speeding up traffic destinated to the bridge itself

In some situations the bridge not only serves as a bridge box, but also talks to other hosts. Packets that arrive on a bridge port and that are destinated to the bridge box itself will by default enter the iptables INPUT chain with the logical bridge port as input device. These packets will be queued twice by the network code, the first time they are queued after they are received by the network device. The second time after the bridge code examined the destination MAC address and determined it was a locally destinated packet and therefore decided to pass the frame up to the higher protocol stack.**[1] (http://ebtables.netfilter.org/examples/basic.html#ex_speed)**

The way to let locally destinated packets be queued only once is by brouting them in the BROUTING chain of the broute table. Suppose br0 has an IP address and that br0's bridge ports do not have an IP address. Using the following rule should make all locally directed traffic be queued only once:

```
# ebtables -t broute -A BROUTING -d $MAC_OF_BR0 -p ipv4 -j redirect --redirect-target DROP
```

The replies from the bridge will be sent out through the br0 device (assuming your routing table is correct and sends all traffic through br0), so everything keeps working neatly, without the performance loss caused by the packet being queued twice.

The redirect target is needed because the MAC address of the bridge port is not necessarily equal to the MAC address of the bridge device. The packets destinated to the bridge box will have a destination MAC address equal to that of the bridge br0, so that destination address must be changed to that of the bridge port.

See also

- **Official documentation for bridge-utils (http://www.linuxfoundation.org/collaborate/workgroups/networking/bridge)**
- **Official documentation for iproute2 (http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2)**
- **ebtables/iptables interaction on a Linux-based bridge (http://ebtables.netfilter.org/br_fw_ia/br_fw_ia.html)**

Retrieved from "https://wiki.archlinux.org/index.php?title=Network_bridge&oldid=497042"