Network configuration

This page explains how to set up a **wired** connection to a network. If you need to set up **wireless** networking see the **Wireless network configuration** page.

Contents

Related articles

Jumbo frames

Firewalls

Wireless network configuration

Network bridge

List of applications/Internet#Network managers

Network Debugging

Check the connection

The basic installation procedure typically has a functional network configuration. Use **ping(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ping.8)** to check the connection:

```
$ ping www.google.com
```

```
PING www.l.google.com (74.125.132.105) 56(84) bytes of data.
64 bytes from wb-in-f105.1e100.net (74.125.132.105): icmp_req=1 ttl=50 time=17.0 ms
...
```

If the ping is successful (you see 64 bytes messages as above), then the network is configured. Press `Control-C` to stop the ping.

If the ping failed with an *Unknown hosts* error, it means that your machine was unable to resolve this domain name. It may be related to your service provider or your router/gateway. Try pinging a static IP address to prove that your machine has access to the Internet:

```
$ ping 8.8.8.8
```

```
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=53 time=52.9 ms
...
```

If you are able to ping `8.8.8.8` but not `www.google.com`, check your DNS configuration. See **resolv.conf** for details. The `hosts` line in `/etc/nsswitch.conf` is another place you can check.

If not, check for cable issues before diagnosing further.

Note:

- If you receive an error like `ping: icmp open socket: Operation not permitted` when executing *ping*, try to re-install the **iputils (https://www.archlinux.org/packages/?name=iputils)** package.
- The `-c num` option can be used to make exactly `num` pings, otherwise it pings infinitely and has to be terminated manually. See **ping(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ping.8)** for more information.
- `8.8.8.8` is a static address that is easy to remember. It is the address of Google's primary DNS server, therefore it can be considered reliable, and is generally not blocked by content filtering systems and proxies.

Device driver

Check the status

udev should detect your **network interface controller** and automatically load the necessary module at start up. Check the "Ethernet controller" entry (or similar) from the `lspci -v` output. It should tell you which kernel module contains the driver for your network device. For example:

```
$ lspci -v
```

```
02:00.0 Ethernet controller: Attansic Technology Corp. L1 Gigabit Ethernet Adapter (rev b0)
        ...
        Kernel driver in use: atl1
        Kernel modules: atl1
```

Next, check that the driver was loaded via `dmesg | grep module_name`. For example:

```
$ dmesg | grep atl1
```

```
...
atl1 0000:02:00.0: eth0 link is up 100 Mbps full duplex
```

Skip the next section if the driver was loaded successfully. Otherwise, you will need to know which module is needed for your particular model.

Load the module

Search in the Internet for the right module/driver for the chipset. Some common modules are `8139too` for cards with a Realtek chipset, or `sis900` for cards with a SiS chipset. Once you know which module to use, try to **load it manually**. If you get an error saying that the module was not found, it's possible that the driver is not included in Arch kernel. You may search the **AUR** for the module name.

If udev is not detecting and loading the proper module automatically during bootup, see **Kernel modules#Automatic module handling**.

Network management

Network interfaces

For computers with multiple NICs, it is important to have fixed interface names. Many configuration problems are caused by interface name changing.

udev is responsible for assigning names to each device. Systemd uses **Predictable Network Interface Names (http://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames)**, which automatically assigns static names to network devices. Interfaces are now prefixed with `en` (wired/**Ethernet**), `wl` (wireless/WLAN), or `ww` (**WWAN**) followed by an automatically generated identifier, creating an entry such as `enp0s25`.

Get current interface names

Both wired and wireless interface names can be found via `ls /sys/class/net` or `ip link`. Note that `lo` is the **loop device** and not used in making network connections.

Wireless device names can also be retrieved using `iw dev`. See also **Wireless network configuration#Get the name of the interface**.

Tip: To change the device names, see **#Change device name** and **#Revert to traditional device names**.

Enabling and disabling network interfaces

You can activate a network interface using:

```
# ip link set interface up
```

To deactivate it do:

```
# ip link set interface down
```

To check the result for the interface `eth0`:

```
$ ip link show dev eth0
```

```
2: eth0: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast master br0 state UP mode DEFAULT qlen 1000
...
```

Note: If your default route is through interface `eth0`, taking it down will also remove the route, and bringing it back up will not automatically reestablish the default route. See **#Manual assignment** for reestablishing it.

Dynamic IP address

See **#Network managers** for a list of options in setting a dynamic IP address.

Static IP address

A static IP address can be configured with most standard **network managers**. Independently of the tool you choose, you will probably need to be prepared with the following information:

- Static IP address

- Subnet mask, or possibly its **CIDR notation**, for example `/24` is the CIDR notation of `255.255.255.0` netmask.
- **Broadcast address**
- **Gateway**'s IP address
- Name server (DNS) IP addresses. See also **resolv.conf**.

If you are running a private network, it is safe to use IP addresses in `192.168.*.*` for your IP addresses, with a subnet mask of `255.255.255.0` and a broadcast address of `192.168.*.255`. The gateway is usually `192.168.*.1` or `192.168.*.254`.

Warning:

- Make sure manually assigned IP addresses do not conflict with DHCP assigned ones. See **this forum thread (http://www.raspberrypi.org/forums/viewtopic.php?f=28&t=16797)**.
- If you share your Internet connection from a Windows machine without a router, be sure to use static IP addresses on both computers to avoid LAN problems.

Tip: Addresses can be calculated with the **`ipcalc (https://www.archlinux.org/packages/?name=ipcalc)`** package; see **#Calculating addresses**.

Manual assignment

It is possible to manually set up a static IP using only the **`iproute2 (https://www.archlinux.org/packages/?name=iproute2)`** package. This is a good way to test connection settings since the connection made using this method will not persist across reboots. First enable the **network interface**:

```
# ip link set interface up
```

Assign a static IP address in the console:

```
# ip addr add IP_address/subnet_mask broadcast broadcast_address dev interface
```

Then add your gateway IP address:

```
# ip route add default via default_gateway
```

For example:

```
# ip link set eth0 up
# ip addr add 192.168.1.2/24 broadcast 192.168.1.255 dev eth0
# ip route add default via 192.168.1.1
```

Tip: If you get the message `RTNETLINK answers: Network is unreachable`, try to break up the route creation in the following two parts:

```
# ip route add 192.168.1.1 dev eth0
# ip route add default via 192.168.1.1 dev eth0
```

To undo these steps (e.g. before switching to a dynamic IP), first remove any assigned IP address:

```
# ip addr flush dev interface
```

Then remove any assigned gateway:

```
# ip route flush dev interface
```

And finally disable the interface:

```
# ip link set interface down
```

For more options, see the **ip(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip.8)**. These commands can be automated using scripts and **systemd units**.

Calculating addresses

You can use `ipcalc` provided by the **ipcalc (https://www.archlinux.org/packages/?name=ipcalc)** package to calculate IP broadcast, network, netmask, and host ranges for more advanced configurations. An example is using Ethernet over Firewire to connect a Windows machine to Linux. To improve security and organization, both machines have their own network with the netmask and broadcast configured accordingly.

Finding out the respective netmask and broadcast addresses is done with `ipcalc`, by specifying the IP of the Linux NIC `10.66.66.1` and the number of hosts (here two):

```
$ ipcalc -nb 10.66.66.1 -s 1

Address:   10.66.66.1

Netmask:   255.255.255.252 = 30
Network:   10.66.66.0/30
HostMin:   10.66.66.1
HostMax:   10.66.66.2
Broadcast: 10.66.66.3
Hosts/Net: 2                    Class A, Private Internet
```

Network managers

There are many solutions to choose from, but remember that all of them are mutually exclusive; you should not run two daemons simultaneously. The following table compares the different connection managers. *Automatically handles wired connection* means that there is at least one option for the user to simply start the daemon without creating a configuration file.

Connection manager	Automatically handles wired connection	Official GUI	Archiso [1] (https://git.archlinux.org/archiso.git/tree/configs/releng/packages.both)	Console tools	Systemd units
ConnMan	Yes	No	No	`connmanctl`	`connman.service`
dhcpcd	Yes	No	Yes (`base` (https://www.archlinux.org/groups/x86_64/base/))	`dhcpcd`	`dhcpcd.service`, `dhcpcd@interface.service`
netctl	Yes	No	Yes (`base` (https://www.archlinux.org/groups/x86_64/base/))	`netctl`	`netctl-ifplugd@interface.service`
NetworkManager	Yes	Yes	No	`nmcli`, `nmtui`	`NetworkManager.service`
systemd-networkd	No	No	Yes (`base` (https://www.archlinux.org/groups/x86_64/base/))		`systemd-networkd.service`, `systemd-resolved.service`
Wicd	Yes	Yes	No	`wicd-curses`	`wicd.service`

See also **List of applications#Network managers**.

Set the hostname

A **hostname** is a unique name created to identify a machine on a network, configured in `/etc/hostname` —see `hostname(5)` (https://jlk.fjfi.cvut.cz/arch/manpages/man/hostname.5) and `hostname(7)` (https://jlk.fjfi.cvut.cz/arch/manpages/man/hostname.7) for details. The file can contain the system's domain name, if any. To set the hostname, **edit** `/etc/hostname` to include a single line with *`myhostname`*:

```
/etc/hostname
```

```
myhostname
```

Tip: For advice on choosing a hostname, see **RFC 1178 (https://tools.ietf.org/html/rfc1178)**.

Alternatively, using **hostnamectl(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/hostnamectl.1)**:

```
# hostnamectl set-hostname myhostname
```

To temporarily set the hostname (until reboot), use **hostname(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/hostname.1)** from **inetutils (https://www.archlinux.org/packages/?name=inetutils)**:

```
# hostname myhostname
```

To set the "pretty" hostname and other machine metadata, see **machine-info(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/machine-info.5#https%3A%2F%2Fwww.freedesktop.org%2Fsoftware%2Fsystemd%2Fman%2Fmachine-info.html)**.

Local network hostname resolution

The pre-requisite is to **#Set the hostname**, after which hostname resolution works on the local system itself:

```
$ ping myhostname
```

```
PING myhostname (192.168.1.2) 56(84) bytes of data.
64 bytes from myhostname (192.168.1.2): icmp_seq=1 ttl=64 time=0.043 ms
```

To allow other machines to address the host by name, it is necessary to either:

- Configure the **hosts(5) (https://jlk.fjfi.cvut.cz/arch/manpages/man/hosts.5)** file, or
- Enable a service which resolves the hostname.

Note: **systemd (https://www.archlinux.org/packages/?name=systemd)** provides hostname resolution via the `myhostname` nss module, enabled by default in `/etc/nsswitch.conf`. However, clients may still rely on `/etc/hosts`, see **[2] (https://lists.debian.org/debian-devel/2013/07/msg00809.html) [3] (https://bugzilla.mozilla.org/show_bug.cgi?id=87717#c55)** for examples.

To configure the hosts file, add the following line to `/etc/hosts`:

```
127.0.1.1       myhostname.localdomain  myhostname
```

As a result the system resolves to both entries:

```
$ getent hosts
```

```
127.0.0.1       localhost
127.0.1.1       myhostname.localdomain  myhostname
```

For a system with a permanent IP address, that permanent IP address should be used instead of `127.0.1.1`.

Note: Another option is to set up a full DNS server such as **BIND** or **Unbound**, but that is overkill and too complex for most systems. For small networks and dynamic flexibility with hosts joining and leaving the network **zero-configuration networking** services may be more applicable:

- **Samba** provides hostname resolution via Microsoft's **NetBIOS**. It only requires installation of `samba (https://www.archlinux.org/packages/?name=samba)` and enabling of the `nmbd.service` service. Computers running Windows, macOS, or Linux with `nmbd` running, will be able to find your machine.
- **Avahi** provides hostname resolution via **zeroconf**, also known as Avahi or Bonjour. It requires slightly more complex configuration than Samba: see **Avahi#Hostname resolution** for details. Computers running macOS, or Linux with an Avahi daemon running, will be able to find your machine. Windows does not have an built-in Avahi client or daemon.

Tips and tricks

Change device name

Note: When changing the naming scheme, do not forget to update all network-related configuration files and custom systemd unit files to reflect the change.

You can change the device name by defining the name manually with an udev-rule. For example:

`/etc/udev/rules.d/10-network.rules`

```
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="aa:bb:cc:dd:ee:ff", NAME="net1"
SUBSYSTEM=="net", ACTION=="add", ATTR{address}=="ff:ee:dd:cc:bb:aa", NAME="net0"
```

These rules will be applied automatically at boot.

A couple of things to note:

- To get the MAC address of each card, use this command: `cat /sys/class/net/device_name/address`
- Make sure to use the lower-case hex values in your udev rules. It doesn't like upper-case.

If the network card has a dynamic MAC, you can use `DEVPATH`, for example:

/etc/udev/rules.d/10-network.rules

```
SUBSYSTEM=="net", DEVPATH=="/devices/platform/wemac.*", NAME="int"
SUBSYSTEM=="net", DEVPATH=="/devices/pci*/*1c.0/*/net/*", NAME="en"
```

The device path should match both the new and old device name, since the rule may be executed more than once on bootup. For example, in the second rule, `"/devices/pci*/*1c.0/*/net/enp*"` would be wrong since it will stop matching once the name is changed to `en`. Only the system-default rule will fire the second time around, causing the name to be changed back to e.g. `enp1s0`.

To **test** your rules, they can be triggered directly from userspace, e.g. with `udevadm --debug test /sys/DEVPATH`. Remember to first take down the interface you are trying to rename (e.g. `ip link set enp1s0 down`).

Note: When choosing the static names **it should be avoided to use names in the format of "eth*X*" and "wlan*X*"**, because this may lead to race conditions between the kernel and udev during boot. Instead, it is better to use interface names that are not used by the kernel as default, e.g.: `net0`, `net1`,

`wifi0`, `wifi1`. For further details please see the **systemd (http://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames)** documentation.

Revert to traditional device names

If you would prefer to retain traditional interface names such as eth0, **Predictable Network Interface Names (http://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames)** can be disabled by masking the udev rule:

```
# ln -s /dev/null /etc/udev/rules.d/80-net-setup-link.rules
```

Alternatively, add `net.ifnames=0` to the **kernel parameters**.

Set device MTU and queue length

You can change the device **MTU** and queue length by defining manually with an udev-rule. For example:

/etc/udev/rules.d/10-network.rules

```
ACTION=="add", SUBSYSTEM=="net", KERNEL=="wl*", ATTR{mtu}="1500", ATTR{tx_queue_len}="2000"
```

Note:

- `mtu`: For PPPoE, the MTU should be no larger than 1492. You can also set MTU via `systemd.netdev(5)` (https://jlk.fjfi.cvut.cz/arch/manpages/man/systemd.netdev.5).

- `tx_queue_len` : Small value for slower devices with a high latency like modem links and ISDN. High value is recommend for server connected over the high-speed Internet connections that perform large data transfers.

ifplugd for laptops

Tip: **dhcpcd** provides the same feature out of the box.

ifplugd (https://www.archlinux.org/packages/?name=ifplugd) is a daemon which will automatically configure your Ethernet device when a cable is plugged in and automatically unconfigure it if the cable is pulled. This is useful on laptops with onboard network adapters, since it will only configure the interface when a cable is really connected. Another use is when you just need to restart the network but do not want to restart the computer or do it from the shell.

By default it is configured to work for the `eth0` device. This and other settings like delays can be configured in `/etc/ifplugd/ifplugd.conf` .

Note: **netctl** package includes `netctl-ifplugd@.service` , otherwise you can use `ifplugd@.service` from **ifplugd (https://www.archlinux.org/packages/?name=ifplugd)** package. For example, **enable** `ifplugd@eth0.service` .

Bonding or LAG

See **netctl#Bonding** or **Wireless bonding**.

IP address aliasing

IP aliasing is the process of adding more than one IP address to a network interface. With this, one node on a network can have multiple connections to a network, each serving a different purpose. Typical uses are virtual hosting of Web and FTP servers, or reorganizing servers without having to update any other machines (this is especially useful for nameservers).

Example

To manually set an alias, for some NIC, use **iproute2 (https://www.archlinux.org/packages/?name=iproute2)** to execute

```
# ip addr add 192.168.2.101/24 dev eth0 label eth0:1
```

To remove a given alias execute

```
# ip addr del 192.168.2.101/24 dev eth0:1
```

Packets destined for a subnet will use the primary alias by default. If the destination IP is within a subnet of a secondary alias, then the source IP is set respectively. Consider the case where there is more than one NIC, the default routes can be listed with `ip route`.

Change MAC/hardware address

See **MAC address spoofing**.

Internet sharing

See **Internet sharing**.

Router configuration

See **Router**.

Promiscuous mode

Toggling **promiscuous mode** will make a (wireless) NIC forward all traffic it receives to the OS for further processing. This is opposite to "normal mode" where a NIC will drop frames it is not intended to receive. It is most often used for advanced network troubleshooting and **packet sniffing**.

/etc/systemd/system/promiscuous@.service

```
[Unit]
Description=Set %i interface in promiscuous mode
After=network.target

[Service]
Type=oneshot
ExecStart=/usr/bin/ip link set dev %i promisc on
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

If you want to enable promiscuous mode on interface `eth0` run **enable** `promiscuous@eth0.service`.

Troubleshooting

Swapping computers on the cable modem

Some cable ISPs (videotron for example) have the cable modem configured to recognize only one client PC, by the MAC address of its network interface. Once the cable modem has learned the MAC address of the first PC or equipment that talks to it, it will not respond to another MAC address in any way. Thus if you swap one PC for another (or for a router), the new PC (or router) will not work with the cable modem, because the new PC (or router) has a MAC address different from the old one. To reset the cable modem so that it will recognise the new PC, you must power the cable modem off and on again. Once the cable modem has rebooted and gone fully online again (indicator lights settled down), reboot the newly connected PC so that it makes a DHCP request, or manually make it request a new DHCP lease.

If this method does not work, you will need to clone the MAC address of the original machine. See also **#Change MAC/hardware address**.

The TCP window scaling problem

TCP packets contain a "window" value in their headers indicating how much data the other host may send in return. This value is represented with only 16 bits, hence the window size is at most 64Kb. TCP packets are cached for a while (they have to be reordered), and as memory is (or used to be) limited, one host could easily run out of it.

Back in 1992, as more and more memory became available, **RFC 1323 (http://www.faqs.org/rfcs/rfc1323.html)** was written to improve the situation: Window Scaling. The "window" value, provided in all packets, will be modified by a Scale Factor defined once, at the very beginning of the connection. That 8-bit Scale Factor allows the Window to be up to 32 times higher than the initial 64Kb.

It appears that some broken routers and firewalls on the Internet are rewriting the Scale Factor to 0 which causes misunderstandings between hosts. The Linux kernel 2.6.17 introduced a new calculation scheme generating higher Scale Factors, virtually making the aftermaths of the broken routers and firewalls more visible.

The resulting connection is at best very slow or broken.

How to diagnose the problem

First of all, let's make it clear: this problem is odd. In some cases, you will not be able to use TCP connections (HTTP, FTP, ...) at all and in others, you will be able to communicate with some hosts (very few).

When you have this problem, the `dmesg`'s output is OK, logs are clean and `ip addr` will report normal status... and actually everything appears normal.

If you cannot browse any website, but you can ping some random hosts, chances are great that you're experiencing this problem: ping uses ICMP and is not affected by TCP problems.

You can try to use **Wireshark**. You might see successful UDP and ICMP communications but unsuccessful TCP communications (only to foreign hosts).

Ways of fixing it

Bad

To fix it the bad way, you can change the `tcp_rmem` value, on which Scale Factor calculation is based. Although it should work for most hosts, it is not guaranteed, especially for very distant ones.

```
# echo "4096 87380 174760" > /proc/sys/net/ipv4/tcp_rmem
```

Good

Simply disable Window Scaling. Since Window Scaling is a nice TCP feature, it may be uncomfortable to disable it, especially if you cannot fix the broken router. There are several ways to disable Window Scaling, and it seems that the most bulletproof way (which will work with most kernels) is to add the following line to `/etc/sysctl.d/99-disable_window_scaling.conf` (see also **sysctl**):

```
net.ipv4.tcp_window_scaling = 0
```

Best

This problem is caused by broken routers/firewalls, so let's change them. Some users have reported that the broken router was their very own DSL router.

More about it

This section is based on the LWN article **TCP window scaling and broken routers (http://lwn.net/Articles/92727/)** and a Kernel Trap article: **Window Scaling on the Internet (http://kerneltrap.org/node/6723)**.

There are also several relevant threads on the LKML.

Realtek no link / WOL problem

Users with Realtek 8168 8169 8101 8111(C) based NICs (cards / and on-board) may notice a problem where the NIC seems to be disabled on boot and has no Link light. This can usually be found on a dual boot system where Windows is also installed. It seems that using the official Realtek drivers (dated anything after May 2007) under Windows is the cause. These newer drivers disable the Wake-On-LAN feature by disabling the NIC at Windows shutdown time, where it will remain disabled until the next time Windows boots. You will be able to notice if this problem is affecting you if the Link light remains off until Windows boots up; during Windows shutdown the Link light will switch off. Normal operation should be that the link light is always on as long as the system is on, even during POST. This problem will also affect other operating systems without newer drivers (eg. Live CDs). Here are a few fixes for this problem.

Enable the NIC directly in Linux

Follow **#Enabling and disabling network interfaces** to enable the interface.

Rollback/change Windows driver

You can roll back your Windows NIC driver to the Microsoft provided one (if available), or roll back/install an official Realtek driver pre-dating May 2007 (may be on the CD that came with your hardware).

Enable WOL in Windows driver

Probably the best and the fastest fix is to change this setting in the Windows driver. This way it should be fixed system-wide and not only under Arch (eg. live CDs, other operating systems). In Windows, under Device Manager, find your Realtek network adapter and double-click it. Under the "Advanced" tab, change "Wake-on-LAN after shutdown" to "Enable".

In Windows XP (example):

```
Right click my computer and choose "Properties"
--> "Hardware" tab
  --> Device Manager
    --> Network Adapters
      --> "double click" Realtek ...
        --> Advanced tab
          --> Wake-On-Lan After Shutdown
            --> Enable
```

Note: Newer Realtek Windows drivers (tested with *Realtek 8111/8169 LAN Driver v5.708.1030.2008*, dated 2009/01/22, available from GIGABYTE) may refer to this option slightly differently, like *Shutdown Wake-On-LAN --> Enable*. It seems that switching it to `Disable` has no effect (you will notice the Link light still turns off upon Windows shutdown). One rather dirty workaround is to boot to Windows and just reset the system (perform an ungraceful restart/shutdown) thus not giving the Windows driver a chance to disable LAN. The Link light will remain on and the LAN adapter will remain accessible after POST - that is until you boot back to Windows and shut it down properly again.

Newer Realtek Linux driver

Any newer driver for these Realtek cards can be found for Linux on the realtek site (untested but believed to also solve the problem).

Enable *LAN Boot ROM* in BIOS/CMOS

It appears that setting *Integrated Peripherals --> Onboard LAN Boot ROM --> Enabled* in BIOS/CMOS reactivates the Realtek LAN chip on system boot-up, despite the Windows driver disabling it on OS shutdown.

Note: This was tested several times on a GIGABYTE GA-G31M-ES2L motherboard, BIOS version F8 released on 2009/02/05.

No interface with Atheros chipsets

Users of some Atheros ethernet chips are reporting it does not work out-of-the-box (with installation media of February 2014). The working solution for this is to install **backports-patched (https://aur.archlinux.org/packages/backports-patched/)**[AUR].

Broadcom BCM57780

This Broadcom chipset sometimes does not behave well unless you specify the order of the modules to be loaded. The modules are `broadcom` and `tg3`, the former needing to be loaded first.

These steps should help if your computer has this chipset:

- Find your NIC in *lspci* output:

```
$ lspci | grep Ethernet
```

```
02:00.0 Ethernet controller: Broadcom Corporation NetLink BCM57780 Gigabit Ethernet PCIe (rev 01)
```

- If your wired networking is not functioning in some way or another, try unplugging your cable then doing the following:

```
# modprobe -r tg3
# modprobe broadcom
# modprobe tg3
```

- Plug your network cable in. If this solves your problems you can make this permanent by adding `broadcom` and `tg3` (in this order) to the `MODULES` array in `/etc/mkinitcpio.conf`:

```
MODULES=".. broadcom tg3 .."
```

- Rebuild the initramfs:

```
# mkinitcpio -p linux
```

- Alternatively, you can create an `/etc/modprobe.d/broadcom.conf`:

```
softdep tg3 pre: broadcom
```

Note: These methods may work for other chipsets, such as BCM57760.

Realtek RTL8111/8168B

```
# lspci | grep Ethernet
```

```
03:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168B PCI Express Gigabit Ethernet controller (rev 02)
```

The adapter should be recognized by the `r8169` module. However, with some chip revisions the connection may go off and on all the time. The alternative **r8168 (https://www.archlinux.org/packages/?name=r8168)** should be used for a reliable connection in this case. **Blacklist** `r8169`, if **r8168 (https://www.archlinux.org/packages/?name=r8168)** is not automatically loaded by **udev**, see **Kernel modules#Automatic module handling**.

Another fault in the drivers for some revisions of this adapter is poor IPv6 support. **IPv6#Disable functionality** can be helpful if you encounter issues such as hanging webpages and slow speeds.

Gigabyte Motherboard with Realtek 8111/8168/8411

With motherboards such as the Gigabyte GA-990FXA-UD3, booting with IOMMU off (which can be the default) will cause the network interface to be unreliable, often failing to connect or connecting but allowing no throughput. This will apply not only to the onboard NIC, but any other pci-NIC you put in the box because the IOMMU setting affects the entire network interface on the board. Enabling IOMMU and booting with the install media will throw AMD I-10/xhci page faults for a second, but then boot normally, resulting in a fully functional onboard NIC (even with the r8169 module).

When configuring the boot process for your installation, add `iommu=soft` as a **kernel parameter** to eliminate the error messages on boot and restore USB3.0 functionality.

See also

- **Debian Reference: Network setup (https://www.debian.org/doc/manuals/debian-reference/ch05.en.html)**
- **RHEL7: Networking Guide (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/)**
- **Linux Home Networking (http://www.linuxhomenetworking.com/wiki/)**
- **Monitoring and tuning the Linux Networking Stack: Receiving data (https://blog.packagecloud.io/eng/2016/06/22/monitoring-tuning-linux-networking-stack-receiving-data/)**
- **Monitoring and tuning the Linux Networking Stack: Sending data (https://blog.packagecloud.io/eng/2017/02/06/monitoring-tuning-linux-networking-stack-sending-data/)**
- **Tracing a packet journey using tracepoints, perf and eBPF (http://blog.yadutaf.fr/2017/07/28/tracing-a-packet-journey-using-linux-tracepoints-perf-ebpf/)**