USB Ethernet Gadget

A Beginner's Guide

Table of Contents

1 Colophon

2 Introduction

This is a guide to using the USB ethernet gadget functionality on Raspberry Pi models that support it.

It is assumed that the reader has a basic familiarity with the Linux command line and at least one text editor. Desktop users will need to open a terminal to execute many of the commands in this guide.

2.1 What's Not Covered

- Advance networking configurations.
- IPv6
- Operating systems on the USB host other than Raspberry Pi OS.
- Operating systems on the zero other than Raspberry Pi OS lite.
- Desktop network configuration tools.
- Booting over USB and using an NFS root over the USB Ethernet gadget[1].
- Compute modules.

1 It's possible but non-trivial.

2.2 Requirements:

- Raspberry Pi (any model) and the normal accessories to act as the USB host.
- A Raspberry Pi zero, zeroW, zeroWH, zero2W, A, A+, 3A+, 4B, 400, or Pi5 and the normal accessories.
- A user account on both Pi with permission to use `sudo` or the ability to login as root.
- Knowledge if which named release of RPiOS your are running `head -1 /etc/os-release` will tell you.
- An appropriate USB cable:
  - zero, zeroW, zeroWH, zero2W: USB A male to micro B male charge and sync.
  - A, A+, 3A+: USB A male to A male.
  - 4B, 400, Pi5: USB C male to USB A male.
- Optional but highly recommended for troubleshooting:
  - zero, zeroW, and zero2W: GPIO headers fitted.
  - Three female to female jumper (Dupont) wires.
  - A second channel into the Pi running in gadget mode.
    - zeroW(H), 3A+: the onboard WiFi or serial port.
    - 4B, 400, Pi5: the onboard WiFi, onboard Ethernet, or serial port.
    - Others: serial port.

2.3 Conventions

```
Text like this indicates input to or output from the command line.
```

`Text like this` also refers to full or partial commands but is not generally intended to be entered into the command line as is.

“SD card” refers equally to full size and micro SD cards.

“zero” refers to the Pi running the ethernet gadget. Where other models require different configurations this will be noted.

“usb0” refers to the interface(s) provided by the ethernet gadget. The actual interface name may be different. Adjust accordingly when editing configuration files.

3 The Ethernet Gadget

3.1 What it Does Do

- Provide a network interface to the operating system on the zero.
- Provide a network interface to the operating system on the USB host.
- Transfer data between these over the USB link.

3.2 What it Does Not Do

- Provide drivers to either OS
- Anything at layer 3 or above in the OSI model[2]: routing, TCP/IP, DNS, DHCP, etc.

3.3 Restrictions

The ethernet gadget cannot be used with:

- Any PI model that has more than one onboard USB port (4B excepted).
- A USB hub[3].
- Any single USB port Pi model that boots from a USB device[4].

The 4B only supports running the ethernet gadget on its USB C port however there are potential issues with this:

- If your USB host cannot provide sufficient current for the 4B it will need to have its own PSU. This can be achieved by either powering it via the GPIO header or a suitable Y cable.
- When using a separate PSU for the 4B and the USB host there is a risk of back powering the host. Use a modified USB cable or adapter with the +ve line cut to avoid this.
- While the connector is USB C, the 4B only provides a USB 2 connection and does not support USB PD.
- “E marked” cables are known to cause problems with early revision 4Bs.

The USB A female ports on a 4B may be used freely and do not affect the USB C port.

2 See https://en.wikipedia.org/wiki/OSI_model
3 Including any HATs or HAT like devices that connect via test points to provide additional USB ports.
4 Including via USB network adapters.

4 Configure The Zero

This can be done either by logging in to the zero or by mounting its SD card in a reader attached to the USB host. All paths below assume logging in to the zero, adjust as needed if using the latter method.

4.1 Select MAC[5] Addresses

While this step can be skipped when using g_ether it is not recommended to do so.

Every network interface must have its own MAC address. Each MAC address must be unique, at least on your network.

Fortunately there is a mechanism to allow locally administered MAC addresses (as opposed to universally administered ones – those set in the hardware during manufacture). Using this should avoid any conflict with commercially produced devices.

Each MAC address consists of six groups of two hexadecimal digits[6] separated by "-" or ":" for example `11:22:33:44:55:66`

One of `2`, `6`, `A`, or `E` in the second digit indicates a locally administered address for example `1A:22:33:44:55:66` All digits except the second may be freely chosen subject to meeting the requirement for a complete address to be unique.

The ethernet gadget provides two interfaces (one to the device running the gadget and one to the USB host) so needs two MAC addresses. Suitable values might be: `12:22:33:44:55:66` and `16:22:33:44:55:66`

4.2 All Methods

1. Back up your config.txt:

   ```
   sudo cp /boot/config.txt /boot/config.bak
   ```

2. Using your preferred text editor add the following to the end of `/boot/config.txt`

   ```
   [all]

   dtoverlay=dwc2,dr_mode=peripheral
   ```

On next boot your OS will use the dwc2 driver in the correct mode to support operation as a USB gadget.

`,dr_mode=peripheral` is optional on zero, zeroW(H), and the 4B[7]. It is mandatory on A, A+, and 3A+ as theses are hardwired to host mode and the ID pin is not present on their USB A connectors.

5 **M**edia **A**ccess **C**ontrol. See https://en.wikipedia.org/wiki/MAC_address
6 Base 16. Digits 0 to 9, letters A to F.
7 The 4B has no ID pin but defaults to device mode.

4.3 Using The g_ether Module

4.3.1 Why

- It’s easy.
- It takes care of the majority of the set up and configuration for you.

4.3.2 Why Not

- It’s old.
- It’s deprecated and will be going away at some point.
- It only provides the ethernet gadget. If you want other gadgets at the same time this isn’t possible.
- In its simplest configuration it generates new, random MAC addresses for the interfaces each time it is started. This will be problematic when getting IP addresses from a DHCP server. It will also cause some OS[8] running on the USB host to treat it as a new interface on each connection.

4.3.3 How

1. Apply the steps in section 4.2
2. Back up `/boot/cmdline.txt`

   ```
   sudo cp /boot/cmdline.txt /boot/cmdline.bak
   ```

3. Using you preferred text editor add the following to the end of `/boot/cmdline.txt`

   ```
   modules-load=dwc2,g_ether g_ether.dev_addr=12:22:33:44:55:66
     g_ether.host_addr=16:22:33:44:55:66
   ```

   Replace the MAC addresses above as required.

   The contents of `/boot/cmdline.txt` must be one a single line.

4. Save
5. Reboot

8 Any Linux with Predictable Network Interface Names enabled and Windows 10 to name two.

4.4 Using The libcomposite Module And configfs

4.4.1 Why

- It gives finer control over the gadget.
- It allows more than one gadget on the same zero.
- When using multiple gadgets they can be of different types[9].

4.4.2 Why Not

- It's more complicated.
- Everything must be setup manually.
- Some USB hosts may initially report an "unknown USB device" during the zero's boot process only to correct this once gadget configuration is complete.

4.4.3 How

This is a step by step guide as would be typed at a terminal. In practise you'll probably want to put this in a script[10].

1. Apply the steps in section 4.2
2. Become root:

```
sudo -i
```

3. Load libcomposite:

```
modprobe libcomposite
```

4. Create a directory for the gadget configuration:

```
mkdir -p /sys/kernel/config/usb_gadget/mygadget
```

5. Make that the current directory:

```
cd /sys/kernel/config/usb_gadget/mygadget
```

6. Set the IDs for the gadget:

```
echo 0x1d6b > idVendor
echo 0x0104 > idProduct
echo 0x0100 > bcdDevice
echo 0x0200 > bcdUSB
```

7. Configure text strings:

9 Using multiple gadgets is outside the scope of this guide.
10 Except the first two and the last one.

```
mkdir -p strings/0x409
echo "1234567890" > strings/0x409/serialnumber
echo "me" > strings/0x409/manufacturer
echo "My USB Device" > strings/0x409/product
```

If you know it, use the zero's serial number. Change the other strings to taste.

8. Initial device configuration:

```
mkdir -p configs/c.1/strings/0x409
echo "Config 1: ECM network" > configs/c.1/strings/0x409/configuration
echo 250 > configs/c.1/MaxPower
```

9. Configure the ethernet gadget:

```
mkdir -p functions/ecm.usb0
echo "12:22:33:44:55:66" > functions/ecm.usb0/host_addr
echo "16:22:33:44:55:66" > functions/ecm.usb0/dev_addr
ln -s functions/ecm.usb0 configs/c.1/
```

Change the two MAC addresses to the ones you decided on in section 4.1

10. Finish up[11]:

```
ls /sys/class/udc > UDC
```

11. Exit the root shell.

11 Additional gadget configuration must be done prior to this step.

4.4.4 Bash Script Example

```
#!/bin/bash
modprobe libcomposite
mkdir -p /sys/kernel/config/usb_gadget/mygadget
cd /sys/kernel/config/usb_gadget/mygadget
echo 0x1d6b > idVendor
echo 0x0104 > idProduct
echo 0x0100 > bcdDevice
echo 0x0200 > bcdUSB
mkdir -p strings/0x409
echo "1234567890" > strings/0x409/serialnumber
echo "me" > strings/0x409/manufacturer
echo "My USB Device" > strings/0x409/product
mkdir -p configs/c.1/strings/0x409
echo "Config 1: ECM network" > configs/c.1/strings/0x409/configuration
echo 250 > configs/c.1/MaxPower
mkdir -p functions/ecm.usb0
echo "12:22:33:44:55:66" > functions/ecm.usb0/host_addr
echo "16:22:33:44:55:66" > functions/ecm.usb0/dev_addr
ln -s functions/ecm.usb0 configs/c.1/
ls /sys/class/udc > UDC
```

1. Save the script
2. Give it execute permission

   ```
   chmod a+x /path/to/script
   ```

3. Call it from root's `crontab` or `/etc/rc.local`.

4.5 Verify It’s Working

If you do not have a second channel into your zero you may wish to delay this step until after networking has been configured (see section 5.1)

1. Boot your USB host.
2. Connect the zero and allow it to boot.
3. Log in to your USB host.
4. Run

   ```
   lsusb
   ```

   The output should contain a line similar to this one:

   ```
   Bus 001 Device 011: ID 0525:a4a2 Netchip Technology, Inc. Linux-USB
      Ethernet/RNDIS Gadget
   ```

5. Run

   ```
   ifconfig -a
   ```

   The output should contain details for a `usb0` interface.

At this stage it will not be possible to communicate over the link as networking has not yet been configured.

5 Basic Network Configuration

5.1 Introduction

Network configuration is handled by the OS not by the ethernet gadget. The IP layer neither knows not cares that it is over USB. Nor does it care which device is the USB master.

This is about as basic as it gets:

- Static IP address for the USB host.
- Static IP address for the zero.
- No communication from the zero to devices other than the USB host over the USB link[12].
- No communication from other devices connected to the same network as the USB host to the zero over the USB link.

5.2 Picking an IP Address Range

There are three address ranges reserved for private networks:

- 10.0.0.0 – 10.255.255.255
- 172.16.0.0 – 172.31.255.255
- 192.168.0.0 – 192.168.255.255

To avoid problems with routing and address conflicts use one of these and stay away from the ranges used by any other interfaces on either the USB host or the zero.

For example, if your router uses 192.168.x.x, use 10.x.x.x for the USB link.

The above ranges can be further divided but as that requires configuration changes to your network it is beyond the scope of this guide.

Once you have decided on an address range, pick two addresses within that range: one for the zero and one for the USB host.

In all the ranges a netmask[13] of 255.255.255.0 is safe to use and provides some leeway if a second point to point link[14] is to be used.

12 This includes the internet.
13 A netmask defines how much of the address is specific to the network and how much to an interface on that network. 255.255.255.0 (sometimes written /24) means the first 3 parts identify the network, the last one the interface.
14 E.g. a second zero running as an ethernet gadget connected to the same USB host.

5.3 Configuring An IP Address

The process is the same for both the zero and the USB host but be sure to use different IP addresses within the same range. For example:

- USB host: 10.0.0.2
- zero: 10.0.0.1
- Netmask: 255.255.255.0 may also be written as a `/24` suffix to the IP address.

Use only one method per device. There is no requirement to use the same method on both the zero and the USB host.

5.3.1 Using dhcpcd.conf

Note: This section applies to RPiOS up to and including Bullseye only.

1. Backup your current configuration:

   ```
   sudo cp /etc/dhcpcd.conf /etc/dhcpcd.bak
   ```

2. Open `/etc/dhcpcd.conf` in your preferred text editor. You will need to be root or use `sudo`.

3. Add the following to the end of the file:

   ```
   interface usb0
      static ip_address=x.x.x.x/y
   ```

   For example:

   ```
   interface usb0
      static ip_address=10.0.0.1/24
   ```

4. Save and close.

5. Repeat for the other device using a different IP address. For example:

   ```
   interface usb0
      static ip_address=10.0.0.2/24
   ```

6. Reboot both devices

The interface name on the zero will almost always be `usb0`. It may be different on the USB host.

For more help and information on `dhcpcd.conf` refer to the comments within the file or the output of

```
man dhcpcd.conf
```

5.3.2 Using /etc/network/interfaces

Using `/etc/network/interfaces` is deprecated and may cause other network interfaces on the device to stop working. It will, however, work on all version of RPiOS up to and including at least Bookworm.

1. Back up your current configuration:

   ```
   sudo cp /etc/network/interfaces /etc/network/interfaces.bak
   ```

2. Open `/etc/network/interfaces` in your preferred text editor. You will need to be root or use `sudo`.

3. Add the following to the end of /etc/network/interfaces:

   ```
   auto usb0
   allow-hotplug usb0
   iface usb0 inet static
   address x.x.x.x
   netmask n.n.n.n
   ```

   Replace x.x.x.x and n.n.n.n with the IP address and netmask you decided on in section 5.2 for example:

   ```
   auto usb0
   allow-hotplug usb0
   iface usb0 inet static
   address 10.0.0.1
   netmask 255.255.255.0
   ```

4. Repeat for the other device using a different IP address. For example:

   ```
   auto usb0
   allow-hotplug usb0
   iface usb0 inet static
   address 10.0.0.2
   netmask 255.255.255.0
   ```

5. Reboot both devices.

The interface name on the zero will almost always be `usb0`. It may be different on the USB host.

For more help and information on `/etc/network/interfaces` see the output of

```
man interfaces
```

5.3.3 libcomposite/configfs

It is possible that the `usb0` interface on the zero may not get an IP address when it has been configured through libcomposite/configfs. In which case it must be brought up manually:

```
sudo ifconfig interface x.x.x.x netmask n.n.n.n up
```

For example:

```
sudo ifconfig usb0 10.0.0.1 netmask 255.255.255.0 up
```

This can be added to the end of your configuration script.

5.3.4 Using Network Manager

Note: RPiOS Bookworm or later. Bullseye and earlier do not use Network Manager unless explicitly configured to do so.

You will need to be root or use sudo to perform these steps.

1. Create a new conection:

   ```
   nmcli con add con-name my-connection type ethernet ifname usb0
      ipv4.method manual ipv4.address x.x.x.x/y
   ```

   Replace `my-connection` with your desired name, `x.x.x.x` with your desired IP address, and `y` with the desired netmask.

   For example:

   ```
   nmcli con add con-name g_ether type ethernet ifname usb0 ipv4.method
      manual ipv4.address 10.0.0.1/24
   ```

2. Activate connection:

   ```
   nmcli con up my-connection
   ```

For more information on `nmcli` see

```
man nmcli
```

For a text menu based alternative try `nmtui`.

6 Troubleshooting

6.1 Configure The Secondary Channel into The Zero

It may be necessary to remove or comment out[15] the `dtoverlay` line from `/boot/config.txt` and reboot in order to allow local login to the zero to perform these steps. Remember to re-enable it afterwards and reboot.

Without a secondary channel into the zero troubleshooting will be several orders of magnitude more difficult.

Once troubleshooting is complete the secondary channel can be disconnected.

6.1.1 Serial Port

Using a serial port allows the boot messages to be monitored, does not impact the network configuration, and does not require reconnection after each reboot.

It requires that the GPIO header has been fitted to the zero and that the UART pins on the USB host are free. A USB to 3.3v[16] serial adapter may be used on the USB host if the pins are not free.

No software configuration to the zero should be required. At time of writing Raspberry Pi OS defaults to having boot messages and a login available on the serial port.

1. Using a jumper wire, connect the TX (GPIO 14, physical 8) pin of the zero to the RX (GPIO 15, physical 10) pin of the USB host.
2. Using a jumper wire, connect the RX (GPIO 15, physical 10) pin of the zero to the TX (GPIO 14, physical 8) pin of the USB host.
3. Connect any ground pin of the zero to any ground pin of the USB host.
4. Login to the USB host.
5. Open `raspi-config`. You'll need to be root or use `sudo`.
6. Ensure a login shell is not accessible over serial. This should be under "Interface Options" then "Serial Port".
7. Ensure the serial port is enabled.
8. Ensure that log in over the serial port is not enabled.
9. Install screen:

```
sudo apt update
sudo apt install screen
```

15 Insert a # at the start of the line.

16 Do not use a 5v adapter, an RS232 adapter, or connect to a PC's nine pin serial port. Doing so will damage the zero.

10. Ensure you user can access the serial ports:

    ```
    sudo usermod -a -G pi dialout
    ```

    Replace pi if using a different user.

11. Reboot.

12. You can now open the serial port:

    ```
    screen /dev/serial0 115200
    ```

Hit `control-a` then `k` to exit.

6.1.2 Network

Zero, A, and A+ models lack onboard networking so cannot use this method.

ZeroW, zeroWH, 3A+, and 4B models can use their onboard WiFi.

The 4B can also use its onboard ethernet.

Using a network connection has a couple of drawbacks:

- You must use a different subnet for this interface to the one being used for the USB link.
- Hostname resolution may not get the IP address you're expecting. Use the IP address when testing the USB link rather than hostnames.

Configuration is straight forward:

1. Set up networking on the zero in the usual way.[17]
2. Enable ssh on the zero.[18]
3. Reboot the zero.

17 See https://www.raspberrypi.org/documentation/configuration/tcpip/README.md and https://www.raspberrypi.org/documentation/configuration/wireless/README.md

18 See https://www.raspberrypi.org/documentation/remote-access/ssh/README.md

6.1.3 Serial Over USB

This needs some changes to your gadget configuration. The connection will also be lost during reboots of the zero and boot messages will not be visible[19].

This is not recommended on Windows USB hosts due to driver issues.

1. Reconfigure the gadget:

   If using g_ether:

   Replace all instances of `g_ether` in `/boot/cmdline.txt` with `g_cdc`

   If using libcomposite/configfs:

   Add the following above the line that reads "`ls /sys/class/udc > UDC`" in your configuration script:

   ```
   mkdir -p functions/acm.usb0
   ln -s functions/acm.usb0 configs/c.1/
   ```

2. Enable login via the gadget serial port:

   ```
   sudo systemctl enable getty@ttyGS0.service
   ```

3. Reboot.

You can now login to the zero with `screen`.[20] Use `/dev/ttyACM0` rather than `/dev/serial0`. If that fails check which serial device has been assigned by the OS.

19 Unlike when using a true serial port.
20 See section 6.1.1 for details.

6.2 No usb0 Interface Found On The USB Host

- Check the cable including any adapters.
- Check you are using the correct cable:

  zero, zeroW, and zeroWH: A male to micro B male.[21]

  A, A+, 3A+: A male to A male

  4B: A male to USB C.
- Check the cable is connected to the correct port on the zero.

  zero, zeroW, and zeroWH: the micro USB port nearest the mini HDMI port.

  A, A+, 3A+: the female USB A port.

  4B: the USB C (power) port.
- A, A+, 3A+ only make sure your /boot/config.txt entry looks like this:

```
dtoverlay=dwc2,dr_mode=peripheral
```

- Check the zero is booting. See section 6.3
- Connect a monitor and watch any boot messages.
- Check your USB host hasn't called it something else:

```
ifconfig -a
```

- Log in to the zero over your secondary channel and check the module(s) have loaded:

```
lsmod
```

- Log in to the zero over your secondary channel and check the `usb0` network interface exists:

```
ifconfig -a
```

- Log in to the zero over your secondary channel and check the logs:

```
more /var/log/syslog
dmesg
```

- Log in to the zero over your secondary channel and try running the configuration manually.

  If using g_ether:

```
sudo modprobe g_ether
```

  If using libcomposite/configfs:

```
sudo /path/to/script
```

21 An OTG adapter is not required and should not be used.

6.3 The Zero Does Not Boot

6.3.1 All cases

- Check the cable including any adapters.
- Connect a monitor and the boot messages, if any.
- Check for errors in `/boot/config.txt`
- Check for errors in `/boot/cmdline.txt`:
  - Its contents must be on a single line.
  - There must be a space between “rootwait” and any additions.
- If using libcomposite/configfs check for errors in your configuration script.
- Follow the usual boot failure trouble shooting steps. See https://www.raspberrypi.org/forums/viewtopic.php?f=28&t=58151
- To rule out hardware failure, try to boot from a different SD card containing a fresh OS installation. If it still fails to boot it’s likely a hardware issue with the zero.

6.3.2 Power From USB Host Only

- Check the USB host’s power supply can provide enough current. See https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md
- 4B gadget only: switch to a “Y” cable[22] (2 x USB A male, 1 x USB C) between the USB host and the 4B or add a separate PSU. The 4B requires a minimum of 600mA on a bare board. This exceeds the 500mA upper limit for a USB 2 device.

6.3.3 Zero With Its Own PSU

Check the power supply can provide enough current. See https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md

Using an unmodified USB cable between the zero and the USB host in this instance may be problematic. If voltages are not exactly the same one will try to feed power to the other.

22 If the USB host is a Raspberry Pi this will make no difference. No current model Pi does per port USB current limiting.

6.4 Cannot Connect By IP Address

- Ensure you are using the correct IP address – the IP address of the `usb0` interface on the target computer.
- Ensure the IP addresses used by the USB link are in a different subnet to that of any other network either computer can see. See section 5.2

6.5 Cannot Connect By Hostname

This is to be expected. With no DNS server neither computer will be able to resolve the hostname of the other to an IP address. If the USB host is connected to a network with a DNS server, it won't help. The DNS server on that network will not know about the zero.

If using static IP address add an entry to `/etc/hosts` on each computer. For example:

```
10.0.0.1    usbgadget
```

It is not usually necessary to add an entry for the local computer, only for the remote one.

6.6 Cannot Connect By Hostname.local

Ensure both computers have zeroconf[23] installed. Raspberry Pi OS has this installed and enabled by default.

6.7 Hostname And/Or Hostname.local Connect to the Wrong Computer

- Ensure you do not have another computer with the same hostname.
- If using entries in /etc/hosts check they are correct and that the IP address of the target computer has not changed.

23 Also known as mDNS, avahi, and bonjour.

6.8 Connecting By Hostname or Hostname.local Reaches The Correct Computer But The Wrong Interface.

This will only happen if both the zero and the USB host are connected to the same network as well as by the ethernet gadget.

- If using entries in /etc/hosts check they are correct and that they are using the correct IP address.
- Use IP address instead of hostname.
- Disconnect one or both computers from the network.

6.9 Ping Works But ssh Does Not

- Ensure you are trying to connect to the correct computer.
- Ensure ssh has been enabled on the target computer.

6.10 Windows[24] Doesn't Recognise The Zero Or Installs The Wrong Driver

Try using `linux.inf` or `linux-cdc-acm.inf` from here:
https://www.kernel.org/doc/Documentation/usb/

6.11 Interface Name or Connection Name On The USB Host Changes Or Saved Configuration Is Lost

This is usually because the MAC address of the host end of the USB gadget has changed since the USB host last saw it. Refer to Section 4 and configure `host_addr` and `dev_addr`.

24 This is the only Windows help I'm going to provide.

7 Advanced Usage

7.1 Using Link Local IP Addressing

7.1.1 Raspberry Pi OS Bullseye and Earlier

Use with `/etc/dhcpcd.conf` only.

1. Remove all configuration for `usb0` from `/etc/dhcpcd.conf` on the zero.
2. Remove all configuration for `usb0` from `/etc/dhcpcd.conf` on the USB host.
3. Remove any entries you added to `/etc/hosts` on both computers.
4. Reboot both computers.

Both computers should now have IP addresses in the link local subnet and should be reachable by `hostname.local`. Connecting by `hostname` is unlikely to work and the IP addresses are highly likely to change across boots.

Either interface can be forced to use link local only as follow:

1. Remove all configuration for `usb0` from `/etc/dhcpcd.conf`.
2. Insert the following at the end of `/etc/dhcpcd.conf`:

```
interface usb0
nodhcp
```

3. Remove any entries you added to `/etc/hosts` on both computers.
4. Reboot.

7.1.2 Raspberry Pi OS Bookworm and Later

Network Manager does not currently fall back to link local addressing. This applies to both the zero and the USB host.

If the USB host is likely to do so:

Create a new connection or edit an existing one to use configure a static IP address for the zero in the link local subnet (169.254.x.x/16).

Alternatively, create a new connection with `ipv4.method link-local` or edit an existing one to set the ipv4 method to link-local. For example:

```
sudo nmcli con add con-name g_ether type ethernet ifname usb0 ipv4.method link-local
```

or

```
sudo nmcli con mod con-name g_ether ipv4.method link-local
```

As always, replace `g_ether` and `usb0` as required.

Network Manager has support for multiple connections on a single interface and will use them according to their `connection.autoconnect-priority` setting. High values having priority over lower values.

While I have found this to be unreliable, the following code snippet may assist in testing this functionality.

```
sudo nmcli con add con-name g_ether-auto type ethernet ifname usb0 ipv4.method auto connection.autoconnect true connection.autoconnect-priority 100

sudo nmcli con add con-name g_ether-linklocal type ethernet ifname usb0 ipv4.method link-local connection.autoconnect true connection.autoconnect-priority 50

sudo nmcli con reload

sudo nmcli con up g_ether-auto

sudo nmcli con up g_ether-linklocal
```

As always, replace `g_ether-auto`, `g_ether-linklocal`, and `usb0` as required.

7.2 Giving The Zero Access To Your Network And The Internet

This should go without saying: if your USB host is not connected to a network and not connected to the internet this will not grant access to the zero.

It should also go without saying that, even with its own power supply, the zero will not be able to access your network and the internet if the USB host is off or disconnected from that network.

The configuration changes required on both the USB host and the zero may disconnect it from your network so are best performed when logged in via the device's console[25].

7.2.1 Bridged Access

Bridging cannot be used when the USB host's network/internet connection is provided by a WiFi interface in client mode.

Advantages:

- The zero is a full member of your network.
- The zero can get its IP address and configuration from the DHCP server on your network.
- Creating a bridge changes which interface(s) the USB host uses to connect to your network.

Disadvantages:

- The zero is a full member of your network. Any computer on your network can connect to it.
- It doesn't work with a WiFi interface in client mode.

25 I.E. via a directly connected monitor and keyboard

Procedure (RPiOS Bullseye and earlier):

1. Reconfigure `/etc/dhcpcd.conf` on the zero so it get its IP configuration for `usb0` via DHCP.
2. On the USB host:
    1. Remove any manual configuration for its ethernet interface and its `usb0` interface.
    2. As root or with `sudo` open your preferred text editor.
    3. Insert the following:

        ```
        [NetDev]
        Name=br0
        Kind=bridge
        MACAddress=dc:a6:32:e8:e1:d4
        ```

        MACAddress is optional but if used should match the MAC address of the USB host's ethernet interface.
    4. Save the file as `/etc/systemd/network/bridge-br0.netdev`
    5. Close and reopen the text editor
    6. Insert the following:

        ```
        [Match]
        Name=usb0
        Name=eth0
        [Network]
        Bridge=br0
        ```

        Change `usb0` and `eth0` as required.
    7. Save the file as `/etc/systemd/network/bridge-br0.network`
    8. Close the text editor.
    9. Open `/etc/dhcpcd.conf` in your preferred text editor. You will need to be root or use `sudo`.
    10. Add the following to the end of the file adjusting interface names as needed:

        ```
        denyinterfaces eth0 usb0
        interface br0
        ```
    11. Save and close.
    12. Enable systemd-networkd:

        ```
        sudo systemctl enable systemd-networkd
        ```

3. Reboot both computers.

Procedure (RPiOS Bookworm):

1. On the zero:
    1. Delete any existing connection for the gadget's interface:

```
sudo nmcli con down my-connection
sudo nmcli con del my-connection
```

    2. Create a new connection:

```
sudo nmcli con add con-name my-connection type ethernet ifname usb0
   ipv4.method auto
```

    3. Activate it:

```
sudo nmcli con up my-connection
```

2. On the USB host:
    1. Remove any existing connections for the ethernet and gadget interfaces
    2. Create a bridge:

```
sudo nmcli con add type bridge ifname br0
```

    3. Add eth0 to it:

```
sudo ncmli con add type ethernet con-name br0-slave-1 ifname eth0
   master br0
```

    4. Add usb0 to it:

```
sudo ncmli con add type ethernet con-name br0-slave-2 ifname usb0
   master br0
```

    5. Apply any desired configuration to the bridge with `nmtui` or

```
sudo nmcli con mod my-connection [...]
```

3. Reboot both computers.

The zero should now have an IP address in the same subnet as the rest of your network and have access to the internet as well as any other devices on your network.

Devices on your network now have access to the zero.

7.2.2 Routed Access

Advantages:

- It works with a WiFi interface in client mode.
- No changes to the ethernet interface configuration on the USB host.
- The zero is not a full member of your network so devices on your network cannot access the zero.

Disadvantages:

- The zero is not a full member of your network so devices on your network cannot access the zero.

Procedure (RPiOS Buster and earlier):

1. Enable IP routing on the USB host:

```
echo net.ipv4.ip_forward=1 | sudo tee /etc/sysctl.d/routing.conf
```

2. Open /etc/rc.local in your preferred text editor. You will need to be root or use sudo.
3. Add the following above the line that reads “exit 0”

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

4. Save and close.
5. Reboot the USB host and the zero

Procedure (RPiOS Bullseye):

1. Enable IP routing on the USB host:

```
echo net.ipv4.ip_forward=1 | sudo tee /etc/sysctl.d/routing.conf
```

2. Open /etc/rc.local in your preferred text editor. You will need to be root or use sudo.
3. Add the following above the line that reads “exit 0”

```
nft add rule ip nat POSTROUTING oifname "eth0" counter masquerade
```

4. Save and close.
5. Reboot the USB host and the zero

Procedure (RPiOS Bookworm):

1. Remove any existing connection for the gadget interface.
2. Add a shared connection:

```
sudo nmcli con add con-name my-connection type ethernet ifname usb0
   ipv4.method shared
```

3. Activate it:

```
sudo nmcli con up my-connection
```

4. If not already so, configure the zero to use DHCP

Network Manager will configure and start DHCP and DNS servers for you and configure IP forwarding.

7.3 Using Multiple Zeros With One USB Host[26]

This may significantly complicate giving the zeros access to your local network and the internet.

7.3.1 Zeros Network Connecting To USB Host Only

Configure each pair of USB interfaces (zero and the corresponding interface on the USB host) as above but use IP addresses in a different subnet for each pair.

If the first pair uses 10.0.0.0/24, use 10.0.1.0/24 for the second, 10.0.2.0/24 for the third, etc.

7.3.2 Zeros Network Connecting To USB Host And Other Zeros

1. On the USB host, combine all gadget interfaces into a network bridge.
2. Give the bridge a static IP address.
3. Either give `usb0` on each zero a unique static IP in the same subnet as the bridge or run a DHCP server on the bridge and configure the zeros to get their IP addresses from it.

7.3.3 Zeros Network Connecting To USB, Your Network, And The Internet

When using bridged access (see section 7.2.1) configure the zeros as indicated. When configuring the USB host ensure all of its `usbN` interfaces are added to the bridge.

Routed access should work with no additional configuration.

7.4 Using One Zero With Multiple USB Hosts[27]

There are essentially three options:

1. Use link local addressing, zeroconf, and access the zero via hostname.local
2. Run a DHCP and DNS server on the zero (e.g. dnsmasq[28]). Let the USB host get its IP configuration from the DHCP server. The zero will require a static IP address.
3. Configure static IP addresses on the zero and on each USB host.

26 Detailed instructions are OS dependent and outside the scope of this guide.
27 No, not at the same time. Use bridging or routing on the USB host for that.
28 https://thekelleys.org.uk/dnsmasq/doc.html

7.5 Sharing The Zero's WiFi With The USB Host[29]

In essence this is the same as configuring the USB host to provide routed access to the zero.

RPiOS Buster and earlier:

1. Enable IP routing:

```
echo net.ipv4.ip_forward=1 | sudo tee /etc/sysctl.d/routing.conf
```

2. Open /etc/rc.local in your preferred text editor. You will need to be root or use sudo.
3. Add the following above the line that reads "exit 0"

```
iptables -t nat -A POSTROUTING -o usb0 -j MASQUERADE
```

4. Save and close.
5. Reboot the USB host and the zero

RPiOS Bullseye:

1. Enable IP routing:

```
echo net.ipv4.ip_forward=1 | sudo tee /etc/sysctl.d/routing.conf
```

2. Open /etc/rc.local in your preferred text editor. You will need to be root or use sudo.
3. Add the following above the line that reads "exit 0"

```
nft add rule ip nat POSTROUTING oifname "usb0" counter masquerade
```

4. Save and close.
5. Reboot the USB host and the zero

RPiOS Bookworm:

1. Remove any existing connection for the gadget interface.
2. Add a shared connection:

```
3. sudo nmcli con add con-name my-connection type ethernet ifname wlan0
   ipv4.method shared
```

4. Activate it:

```
5. sudo nmcli con up my-connection
```

6. If not already so, configure the USB host to use DHCP

Network Manager will configure and start DHCP and DNS servers for you and configure IP forwarding.

29 The 4B can also share its ethernet connection.

8 Change Log

2024-12-20

Updated for Bookworm and Network Manager.

2022-04-20

Added 7.5 Sharing The Zero’s WiFi With The USB Host (github issue #2)

Updated routing instructions for Bullseye (nftables replaced iptables). (github issue #9)

Fixed typo (github issue #4)

Added zero2W.