Network Boot A Pi Zero Via the USB Ethernet Gadget

Table of Contents

1 Colophon....4
2 Introduction....5
2.1 What's Not Covered....5
2.2 Requirements:....5
2.3 Conventions....6
3 Create The initrd For The Zero....7
4 Configuring The Server....8
4.1 Create directories....8
4.2 Install Packages....8
4.3 Configure Network Interfaces....9
4.4 Configure Routing And NAT....10
4.5 Configure dnsmasq....11
4.6 Configure The NFS Exports....12
4.7 Install The Client OS....13
4.8 Configure rpiboot....14
4.9 Reboot....14
5 Configure The Client OS....15
5.1 config.txt....15
5.2 cmdline.txt....15
5.3 fstab....16
6 Test....17
7 Backup....18
8 Multiple Clients....19
8.1 Serial Number Filter....19
8.1.1 Advantages....19
8.1.2 Disadvantages....19
8.2 Overlay Directories....19
8.2.1 Advantages....19
8.2.2 Disadvantages....19
8.3 Configuration....20
8.3.1 Both Methods....20
8.3.2 Serial Number Filter....21
8.3.3 Overlay Directories....22
9 Clients That Require Different Kernels....23
9.1 32 Bit Kernel And Userland....23
9.2 64 Bit Kernel With 32 Bit Userland....24
9.3 When Using Serial Number Filters....25
9.4 When Using Overlay Directories....25
9.5 The Default Client....25
10 g_ether MAC Addresses....26
11 Switching To A Bridged Network....27
12 Hints And Tips....29
12.1 Kernel Updates....29
12.2 Adding More Client Zeros....29
12.3 Replacing a Client Zero....29

12.4 Static IP Addresses For Clients.....30
12.4.1 Method 1.....30
12.4.2 Method 2.....30
13 Change Log.....31

1 Colophon

2 Introduction

This is a guide to network booting a Pi zero[1] using the USB ethernet gadget[2]. It is not aimed at beginners due to the complexity of the process.

It is assumed that the reader is familiar with the Linux command line, the USB ethernet gadget, 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

- IPv6
- Operating systems on the USB host other than Raspberry Pi OS.
- Operating systems on the zero other than Raspberry Pi OS lite.
- Compute modules.

2.2 Requirements:

- An internet connection.
- Raspberry Pi (any model) and the normal accessories to act as the USB host, NFS server, rpiboot server, and TFTP/PXE server.
- A Raspberry Pi zero, zeroW, zeroWH, A, A+, 3A+, or 4B and the normal accessories.
- A user account on both Pi with permission to use `sudo` or the ability to login as root.
- An appropriate USB cable:
  - zero, zeroW, zeroWH: USB A male to micro B male charge and sync.
  - A, A+, 3A+: USB A male to A male.
  - 4B: USB C male to USB A male.
- Optional but highly recommended for troubleshooting:
  - zero and zeroW: 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: the onboard WiFi, onboard Ethernet, or serial port.
    - Others: serial port.

1 Or any other Pi model that can work as a USB ethernet gadget.
2 If you're not familiar with the ethernet gadget see my guide at https://github.com/thagrol/Guides/blob/main/ethernetgadget.pdf

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.

Where “pi” occurs as a username, replace as appropriate.

“server” refers to the USB host, “client” to the zero(s).

3 Create The initrd For The Zero

This step must be done on the zero. Without an intrd it is not possible to network boot over USB.

1. Flash RPiOS lite on to an SD card.
2. Insert into the zero and boot.
3. Allow the first boot process to run.
4. Login to the zero.
5. Optional but recommended: update installed packages.

   ```
   sudo apt update && sudo apt full-upgrade -y
   ```

6. Open `/etc/initramfs-tools/modules` in your preferred text editor. You will need to be root or use sudo.
7. Add the following to the end of the file:

   ```
   dwc2
   g_ether
    libcomposite
    u_ether
    udc-core
    usb_f_rndis
    usb_f_ecm
   ```

8. Save and close.
9. Generate the initrd:

   ```
   sudo update-initramfs -c -k `uname -r`
   ```

   `-c` Create a new initramfs.

   `-k `uname -r`` Use the same version as the running kernel.

10. Make a note of the file name of the initrd.
11. Shutdown the zero:

    ```
    sudo poweroff
    ```

4 Configuring The Server

The server configuration assumes that it has a network and internet connection and that this is shared with the zero(s) via NAT/routing[3].

Ensure the hostname of the server is not the default “raspberrypi”.

Some of this configuration is not strictly necessary when using only a single zero. However applying all of it will make adding further zeros much easier.

4.1 Create directories

If using external (USB) storage it must be formatted with a linux native file system e.g. ext4. Windows file systems (all FAT variants, NTFS, etc.) will not work correctly due to their lack of support for ownership, permissions, and other features.

The following directories are required:

- `/srv/rpiboot`
- `/srv/clients`
- `/srv/clients/default`

`/srv/clients` must be on a filesystem with sufficient free space to hold at least one complete copy of RPiOS.

4.2 Install Packages

1. Update your apt package lists.

   ```
   sudo apt update
   ```

2. Optional: upgrade installed packages:

   ```
   sudo apt full-upgrade -y
   ```

3. Install the following packages:

   `rpiboot`

   `dnsmasq`

   `nfs-kernel-server`

   ```
   sudo apt install -y rpiboot dnsmasq nfs-kernel-server
   ```

Note: dnsmasq is not required if all devices will be using static IP addresses.

3 Routing works with a WiFi interface in client mode, bridging does not. Routing also has less impact to the host’s existing network configuration.

4.3 Configure Network Interfaces

All steps in this section must be performed as root or with `sudo`.

1. Create `/etc/systemd/network/10-usb-bridge.netdev` with the following contents:

   ```
   [NetDev]
   Name=br0
   Kind=bridge
   ```

2. Create `/etc/systemd/network/10-usb-bridge.network` with the following contents:

   ```
   [Match]
   Name=usb*
   [Network]
   Bridge=br0
   ```

3. Create `/etc/systemd/network/20-br0.network` with the following contents:

   ```
   [Match]
   Name=br0
   [Network]
   Address=10.0.0.1/24
   ```

   When selecting an IP address and netmask be sure to use a subnet that is not in use on any other interface the server may have.

4. Open `/etc/dhcpcd.conf` in your preferred text editor.

5. At the start of the file add:

   ```
   denyinterfaces usb* br0
   ```

   This does not have to be the first line in the file but it **must** be before an interfaces definitions.

6. Save and close.

7. Enable systemd-networkd:

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

8. Reboot

At this point you should have an additional network interface (`br0`) with a static IP address. WiFi and ethernet interfaces should be unaffected.

4.4 Configure Routing And NAT

Skip this step if the host has no connection to another network or you do not want clients to have access to it. Clients unable to reach the internet will likely have incorrect system date and time.

1. Create `/etc/sysctl.d/routed-usb.conf` with the following contents:

   ```
   net.ipv4.ip_forward=1
   ```

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

3. Insert the following above the line that reads `exit 0`

   For Buster and earlier:

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

   For Bullseye:

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

   Replace eth0 with the name of the interface providing the connection.

4. Save and close.

5. Enable NAT now:

   For Buster and earlier:

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

   For Bullseye:

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

4.5 Configure dnsmasq

1. Back up the default `/etc/dnsmasq.conf`
2. Create a new `/etc/dnsmasq.conf` with the following contents:

```
local-service
domain-needed
bogus-priv
no-resolv
no-poll
dhcp-authoritative
interface=br0
no-hosts
filterwin2k
stop-dns-rebind
dhcp-range=10.0.0.2,10.0.0.100,24h
```

3. Restart dnsmasq:

```
sudo systemctl restart dnsmasq
```

Refer to `man dnsmasq` for more details and more configuration options. As before adjust the dhcp-range as necessary.

4.6 Configure The NFS Exports

1. Open `/etc/exports` in you preferred text editor. You will need to be root or use `sudo`.
2. Add the following at the end of the file:

   ```
   /srv/clients   10.0.0.0/24(rw,no_subtree_check,no_root_squash)
   ```

   Replace 10.0.0.0/24 with the subnet chosen in 4.3, for details on the export options see `man exports`.
3. Save and close
4. Reload the exports:

   ```
   sudo exportfs -ra
   ```

The above export is intentionally insecure in order to ease the configuration process.

4.7 Install The Client OS

1. Connect the zero's SD card to the USB host using a suitable card reader.
2. Mount the root partition[4]:

```
sudo mount -o ro /dev/sda2 /mnt
```

3. Adjust device node and mountpoint as necessary.
4. Copy the contents:

```
sudo rsync -axrvH /mnt/* /srv/clients/default/
```

5. Unmount the root partition:

```
sudo umount /mnt
```

6. Mount the boot partition:

```
sudo mount -o ro /dev/sda1 /mnt
```

7. Copy the contents:

```
sudo rsync -axrv /mnt/* /srv/clients/default/boot/
```

8. Unmount the boot partition:

```
sudo umount /mnt
```

4 Desktop users may find the automounter has mounted both partitions

4.8 Configure rpiboot

1. Link the default files:

```
sudo ln -s /srv/clients/default/boot/* /srv/rpiboot/
```

2. Create the following .service file:

```
[Unit]
Description=server for USB booting raspberry Pi
After=Network.target

[Service]
Type=simple
PIDFile=rpiboot.pid
Restart=on-failuret
StandardOutput=file:/tmp/rpiboot.log
StandardError=inherit
ExecStart=/usr/bin/rpiboot -l -o -d /srv/rpiboot

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

3. Save it as `/etc/systemd/system/rpiboot.service`
4. Enable and star the service:

```
sudo systemctl enable rpiboot
```

4.9 Reboot

```
sudo reboot
```

5 Configure The Client OS

These steps must be performed on the server before attempting to boot any clients.

5.1 config.txt

1. Back up `/srv/clients/boot/config.txt`
2. Open `/srv/clients/boot/config.txt` in your preferred text editor. You will need to be root or use `sudo`.
3. Add the following at the end of the file:

```
[all]
## reduce CPU load. Remove if the SD card slot will be used post booting
dtparam=sd_poll_once=on
## enable dwc2 driver in device only mode
dtoverlay=dwc2,dr_mode=peripheral
## specify innitramfs
initramfs initrd.img-4.19.66+ followkernel
```

   Replace `initrd.img-4.19.66+` with the file name from 3.

4. Save and close.

5.2 cmdline.txt

1. Back up `/srv/clients/boot/cmdline.txt`
2. Open `/srv/clients/boot/config.txt` in your preferred text editor. You will need to be root or use `sudo`.
3. Add the following at the end of the file:

```
 ip=dhcp modules-load=dwc2,g_ether
```

4. Replace the existing `root=PARTUUID=something` with

```
root=/dev/nfs nfsroot=10.0.0.1:/srv/clients/default
```

   Replace 10.0..0.1 as necesssary.

5. Remove `rootfstype=ext4`
6. Ensure all contents are on a single line.
7. Save and close.

5.3 fstab

1. Back up `/srv/clients/default/etc/fstab`
2. Open `/srv/clients/default/etc/fstab` in your preferred text editor. You will need to be root or use `sudo`.
3. Remove or comment out (prefix with `#`) the lines for `/` and `/boot`
4. Save and close.

6 Test

You now have a configuration that will boot any SD cardless zero that uses the same kernel as the initrd was built from.

1. Reboot the server.
2. Login to the server.
3. Run

   ```
   tail -f /tmp/rpiboot.log
   ```

   This allows monitoring of the rpiboot process
4. Remove the SD card from the zero.
5. If possible connect the zero to a monitor so the boot process can be observed.
6. Connect the zero to the server via USB[5].

The zero should boot, the server should have a `br0` interface comprised of a single `usb0` interface and the zero should be able to reach your network and the internet.

5 If using a zero, zeroW, or zeroWH do not use an OTG adapter and ensure you connect to the correct port.

7 Backup

Before going further or making customisations to the client OS backup the following:

- `/etc/dhcpcd.conf`
- `/etc/dnsmasq.conf`
- `/etc/exports`
- `/etc/rc.local`
- `/etc/sysctl.d/routed-usb.conf`
- `/etc/systemd/network`
- `/etc/systemd/system/rpiboot.service`
- `/srv/clients`

When backing up `/srv/clients` ensure all permissions are preserved.

8 Multiple Clients

While it may be possible to boot multiple clients from one OS directory doing so will be problematic and is not advised.

There are two ways to address this:

- A serial number filter in config.txt
- Use overlay directories in rpiboot.

8.1 Serial Number Filter

8.1.1 Advantages

- Boot files and the NFS export are tied to a single client[6]
- It doesn't matter if the USB port the client is connected to changes.

8.1.2 Disadvantages

- Boot files and NFS export are tied to a single client
- Swapping out a failed (or upgraded) client require updating `config.txt`
- `config.txt` will grow in size and complexity as new clients are added.
- Tools that make changes to `config.txt` e.g. `raspi-config` will change the shared `config.txt` and therefore affect all clients.

8.2 Overlay Directories

8.2.1 Advantages

- No additional filters in `config.txt` are required.
- Replacing a failed client does not require configuration updates.[7]
- Support is built in to `rpiboot`

8.2.2 Disadvantages

- Boot files and NFS export are tied to a particular USB port.

6 Mostly. Serial numbers are no longer guaranteed to be unique but the odds of getting two the same are small.
7 Unless moving to a model using a different kernel

8.3 Configuration

8.3.1 Both Methods

1. Create a directory to contain the OS for the new client:

```
sudo mkdir -p /srv/clients/client1
```

2. Copy the default client OS:

```
sudo rsync -axrv  /srv/clients/default/* /srv/clients/client1/
```

3. Open `/srv/clients/client1/boot/cmdline.txt` in your preferred text editor. You will need to be root or use `sudo`.
4. Replace `/srv/clients/default` with `/srv/clients/client1`
5. Save and close.
6. Open `/srv/clients/client1/etc/hostname` in your preferred text editor. You will need to be root or use `sudo`.
7. Replace `raspberrypi` with a new hostname, e.g. `client1`.
8. Save and close.
9. Open `/srv/clients/client1/etc/hosts` in your preferred text editor. You will need to be root or use `sudo`.
10. Replace `raspberrypi` with the new hostname.
11. Save and close.

8.3.2 Serial Number Filter

1. Boot the zero and login

   ```
   cat /proc/cpuinfo | grep -i serial
   ```

   Output should be similar to

   ```
   Serial          : 00000000a5c0b802
   ```

2. Shutdown the zero.
3. Login to the USB host/server.
4. Link the boot directory to rpiboot:

   ```
   sudo ln -s /srv/clients/client1/boot /srv/rpiboot/client1
   ```

5. Open `/srv/clients/default/boot/config.txt` in your preferred text editor. You will need to be root or use `sudo`.
6. At the end of the file add the following:

   ```
   [0x00000000a5c0b802]
   os_prefix=client1/
   # additional machine specific options


   [all]
   ```

   The `0x` prefix to the serial number is required[8] as is the trailing `/` in `os_prefix`[9].

7. Save and close.

Rather than including client specific options in `config.txt` an `include`[10] directive may be used, however as neither bootcode.bin nor the EEPROM bootloader support the `include` directive not all `config.txt` items can be set his way.

8 See https://www.raspberrypi.org/documentation/configuration/config-txt/conditional.md
9 See https://www.raspberrypi.org/documentation/configuration/config-txt/boot.md
10 See https://www.raspberrypi.org/documentation/configuration/config-txt/misc.md

8.3.3 Overlay Directories

All steps take place on the server.

1. Shutdown and disconnect all zeros.
2. Stop the rpiboot service:

   ```
   sudo systemctl stop rpiboot
   ```

3. Run

   ```
   sudo rpiboot  -vv -d /srv/clients/default/boot | grep -i path
   ```

4. Connect a zero to the desired USB port.
5. Make a note of the last `Path` shown e.g. `1-1.4`
6. Shutdown and disconnect the zero.
7. Link the client's boot directory into rpiboot:

   ```
   sudo ln -s /srv/clients/client1/boot /srv/rpiboot/1-1.4
   ```

   Replace `1-1.4` as necessary.

8. Restart the rpiboot service:

   ```
   sudo systemctl restart rpiboot
   ```

9 Clients That Require Different Kernels

Not all Pi models capable of booting this way use the same kernel. The initrd must match the kernel.

9.1 32 Bit Kernel And Userland

1. Repeat 3 for one Pi in each of the following groups. Skip any groups not required.

   zero, zeroW, zeroWH, A, and A+

   3A+

   4B and 400

2. Login to the server.
3. Copy the initrd files to /srv/clients/default/boot
4. Open /srv/clients/default/boot/cmdline.txt in your preferred text editor. You will need to be root or use sudo.
5. Remove or comment out any existing initramfs line(s).
6. Add the following, replacing the names of the initrd as necessary:

```
[pi0]
initramfs initrd.img-4.19.66+ followkernel
[pi1]
initramfs initrd.img-4.19.66+ followkernel
[pi3+]
initramfs initrd.img-4.19.66-v7+ followkernel
[pi4]
initramfs initrd.img-4.19.66-v7l+ followkernel
[all]
```

7. Save and close.

9.2 64 Bit Kernel With 32 Bit Userland

This is only possible on the 3A+, 4B, and 400.

Repeat 3 for the following groups and copy the initrds.

- 3A+ with arm_64bit=1 in config.txt
- 4B and 400 with arm_64bit=1 in config.txt

If all machines in a group are using the 64 bit kernel, config.txt becomes

```
[pi3+]
arm_64bit=1
initramfs initrd.img-4.19.66-v8+ followkernel
[pi4]
arm_64bit=1
initramfs initrd.img-4.19.66-v8l+ followkernel
[all]
```

Where a mixture of kernels are required use a combination of a model filter and serial number filters. For example.

```
[pi3+]
arm_64bit=1
initramfs initrd.img-4.19.66-v8+ followkernel
[0x12345654321]
arm_64bit=0
initramfs initrd.img-4.19.66-v7+ followkernel
```

9.3 When Using Serial Number Filters[11]

Either use a model filter as above or configure the initramfs (and arm_64bit) items as part of each serial number filter.

9.4 When Using Overlay Directories[12]

No filters are required. Configure each client’s config.txt directly.

9.5 The Default Client

Configure as in 9.1. All current Pi models support 32 bit kernels.

11 See 8.3.2
12 See 8.3.3

10 g_ether MAC Addresses

Unless otherwise instructed g_ether generates new, random MAC addresses for the interfaces it provides each time it is loaded. This has a number of effects:

- The zero's IP address will change across reboots.[13]
- An IP address cannot be reserved by MAC address on your DHCP server.[14]
- Depending on configuration, the number of devices, and frequency of reboots your DHCP server may run out of leases.
- There is a tiny chance that two interfaces will have the same MAC address.
- As DHCP assigned address cannot be guaranteed it is not possible to restrict access to the client OS to a single IP address.[15]

While irritating, with working DHCP and DNS servers; and/or mDNS[16] none of the above are a problem in practice.

Fixed MAC address can be assigned to g_ether if required but these must be manually managed which quickly becomes unwieldy as the number of zeros increases.

Each instance of g_ether requires two MAC addresses. One for the zero and one for the USB host.

MAC addresses can be fixed by appending the relevant module parameter to the client OS' boot/cmdline.txt. For example:

```
g_ether.dev_addr=02:00:00:00:00:02 g_ether.host_addr=06:00:00:00:00:02
```

Both must be specified. Both must be unique, to each other and to any interfaces present on your network including those of other zeros.

It is recommended to use a MAC address in the locally administered range (an address with `2`, `6`, `A`, or `E` as its second digit) to avoid conflicts with addresses set in other hardware.

13 or reloads of the module
14 Strictly speaking it can but as the MAC address changes your client will likely not get it again. Nor will the DHCP server be free to offer it to other devices.
15 See above.
16 AKA bonjour, zeroconf, avahi, etc.

11 Switching To A Bridged Network

Bridging requires that the USB host be using ethernet to connect to your network

It is assumed that all client zeros and the USB host are getting their IP addresses and configuration via DHCP.

1. Shutdown and disconnect all client zeros.
2. Login to the server.
3. Shutdown and disable dnsmasq:

```
sudo systemctl stop dnsmasq
sudo systemctl disable dnsmasq
```

4. Delete `/etc/sysctl.d/routed-usb.conf`
5. Remove or comment out the iptables line in `/etc/rc.local`
6. Find the MAC address of the ethernet interface:

```
ifconfig eth0 | grep ether
```

   Output should be similar to:

```
ether b8:27:eb:c0:b8:02  txqueuelen 1000  (Ethernet)
```

7. Change the contents of `/etc/systemd/network/10-usb-bridge.netdev` to

```
[NetDev]
Name=br0
Kind=bridge
MACAddress=b8:27:eb:c0:b8:02
```

   Replace b8:27:eb:c0:b8:02 with the MAC address from the previous step.

8. Change the contents of `/etc/systemd/network/10-usb-bridge.network` to

```
[Match]
Name=usb*
Name=eth0
[Network]
Bridge=br0
```

9. Delete `/etc/systemd/network/20-br0.network`

10. Open `/etc/dhcpcd.conf` in your preferred text editor.
11. Change the denyinterfaces line to:

```
denyinterfaces usb* eth0
```

12. If there is any active configuration for `eth0`, make sure it is after the `denyinterfaces` line and update it to apply to `br0` instead of `eth0`.
13. Save and close.
14. Open `/etc/exports` in your preferred text editor.
15. Replace `10.0.0.0/24` with the subnet of your network for example `192.168.1.0/24`
16. Save and close.
17. Find the IP address of the ethernet interface:

```
ifconfig eth0 | grep netmask
```

18. For each installed client OS, edit its `cmdline.txt` and replace 10.0.0.1 with the IP address from the previous step.
19. Reboot
20. Reconnect the zeros.

12 Hints And Tips

12.1 Kernel Updates

A new kernel requires a new initrd as boot will fail if the kernel and initrd do not match.

In theory, the initrd should be updated whenever the kernel is however kernel updates can be blocked by executing the following command on each client:

```
sudo apt-mark hold raspberrypi-kernel raspberrypi-kernel-headers
```

Kernel updates on the server's OS should not impact the clients.

12.2 Adding More Client Zeros

Repeat 8 for each new client. While it is possible to mix both methods of identifying clients it may not be advisable to do so.

12.3 Replacing a Client Zero

No additional configuration is needed unless:

- You're using serial number filters in config.txt (update the serial number).
- You're using overlays in rpiboot and have connected the new client to a different USB port (relink the overlay with the correct path).
- The new client needs a different kernel to the old one (update config.txt to use the correct intird).
- Some combination of the above applies.

12.4 Static IP Addresses For Clients

Using static IP addresses when booting multiple clients from one OS export is not recommended.

12.4.1 Method 1

Use fixed MAC addresses (see 10) and reserve IP addresses by MAC address on your DHCP server.

It is only necessary to reserve addresses for the client's end of the USB link.

12.4.2 Method 2

1. In `cmdline.txt` replace `ip=dhcp` with `ip=client address:server address::netmask` e.g.

   ```
   ip=10.0.0.101:10.0.0.1::255.255.255.0
   ```

   The third field (gateway address) only needs to be specified if the NFS server is on a different subnet.[17]

2. Configure a static IP address for `usb0` in `/etc/dhcpcd.conf`

17 See https://www.kernel.org/doc/Documentation/filesystems/nfs/nfsroot.txt

13 Change Log

2022-04-21

Added configuration for nftables.

Added dtparam=sd_poll_once=on to config.txt for clients.