

FAQ: 5. General

5.1. Q: What are BW, SF, and CR?

A:

BW is bandwidth - width of frequency spectrum that is used for transmission

SF is spreading factor - sets how many chips encode each symbol (2^{SF}); higher SF means longer airtime, greater range and sensitivity, but lower data rate.

CR is coding rate - from: <https://www.thethingsnetwork.org/docs/lorawan/fec-and-code-rate/>

TL;DR: default CR to 5 for good stable links. If it is not a solid link and is intermittent, change to CR to 7 or 8.

Forward Error Correction is a process of adding redundant bits to the data to be transmitted. During the transmission, data may get corrupted by interference (changes from 0 to 1 / 1 to 0). These error correction bits are used at the receivers for restoring corrupted bits.

The Code Rate of a forward error correction expresses the proportion of bits in a data stream that actually carry useful information.

There are 4 code rates used in LoRa:

4/5

4/6

4/7

4/8

For example, if the code rate is 4/7, for every 4 bits of useful information, the coder generates a total of 7 bits of data, of which 3 bits are redundant.

Making the bandwidth 2x wider (from BW125 to BW250) allows you to send 2x more bytes in the same time. Making the spreading factor 1 step lower (from SF10 to SF9) allows you to send 2x more bytes in the same time.

Lowering the spreading factor reduces the receiver's sensitivity, so the link tolerates less noise and has shorter range, in exchange for faster transmission. You could compare this to two people taking in a noisy place (a bar for example). If you're far from each other, you have to talk slow

(SF10), but if you're close, you can talk faster (SF7)

So, it's balancing act between speed of the transmission and resistance to noise.

things network is mainly focused on LoRaWAN, but the LoRa low-level stuff still checks out for any LoRa project

5.2. Q: Do MeshCore clients repeat?

A: No, MeshCore clients do not repeat. This is the core of MeshCore's messaging-first design. This is to avoid devices flooding the air wave and create endless collisions, so messages sent aren't received. Note the emergency-comms trade-off: because MeshCore clients never repeat, the network depends entirely on surviving repeater infrastructure. In a disaster where repeaters lose power, clients fall back to direct radio range only - plan redundant, powered repeaters.

In MeshCore, only repeaters and room server with `set repeat on` repeat.

5.3. Q: What happens when a node learns a route via a mobile repeater, and that repeater is gone?

A: If you used to reach a node through a repeater and the repeater is no longer reachable, the client will send the message using the existing (but now broken) known path, the message will fail after 3 retries, and the app will reset the path and send the message as flood on the last retry by default. This can be turned off in settings. If the destination is reachable directly or through another repeater, the new path will be used going forward. Or you can set the path manually if you know a specific repeater to use to reach that destination.

In the case if users are moving around frequently, and the paths are breaking, they just see the phone client retries and revert to flood to attempt to re-establish a path.

5.4. Q: How does a node discovery a path to its destination and then use it to send messages in the future, instead of flooding every message it sends like Meshtastic?

Routes are stored in sender's contact list. When you send a message the first time, the message first gets to your destination by flood routing. When your destination node gets the message, it will send back a delivery report to the sender with all repeaters that the original message went through. This delivery report is flood-routed back to you the sender and is a basis for future direct path. When you send the next message, the path will get embedded into the packet and be evaluated by repeaters. If the hop and address of the repeater matches, it will retransmit the message, otherwise it will not retransmit, hence minimizing utilization.

[Source](#)

5.5. Q: Do public channels always flood? Do private channels always flood?

A: Yes. Group/public channels are one-to-many broadcasts with no single destination, so there is no specific path to discover - they must flood. Repeaters can however deny flood traffic up to some hop limit, with the `set flood.max` CLI command. Administrators of repeaters get to set the rules of their repeaters.

[Source](#)

5.6. Q: what is the public key for the default public channel?

A: The smartphone app key is in hex:

```
8b3387e9c5cdea6ac9e5edbaa115cd72
```

T-Deck uses the same key but in base64

```
iz0H6cXN6mrJ5e26oRXNcg==
```

The third character is the capital letter 'O', not zero `0`

[Source](#)

5.7. Q: Is MeshCore open source?

A: Most of the firmware is freely available. Everything is open source except the T-Deck firmware and Liam's native mobile apps.

- Firmware repo: <https://github.com/meshcore-dev/MeshCore>

5.8. Q: How can I support MeshCore?

A: Provide your honest feedback on GitHub and on [MeshCore Discord server](#). Spread the word of MeshCore to your friends and communities; help them get started with MeshCore.

Support Liam Cottle's smartphone client development by unlocking the server administration wait gate with in-app purchase

Support Rastislav Vysoky (recrof)'s flasher web site and the map web site development through [PayPal](#) or [Revolut](#)

5.9. Q: How do I build MeshCore firmware from source?

A: See instructions here:

<https://discord.com/channels/826570251612323860/1330643963501351004/1341826372120608769>

Build instructions for MeshCore:

For Windows, first install WSL and Python+pip via: <https://plainenglish.io/blog/setting-up-python-on-windows-subsystem-for-linux-wsl-26510f1b2d80>

(Linux, Windows+WSL) In the terminal/shell:

```
sudo apt update
sudo apt install libpython3-dev
sudo apt install python3-venv
```

Mac: python3 should be already installed.

Then it should be the same for all platforms:

```
python3 -m venv meshcore
cd meshcore && source bin/activate
pip install -U platformio
git clone https://github.com/meshcore-dev/MeshCore.git
cd MeshCore
```

open platformio.ini and in `[arduino_base]` edit the `LORA_FREQ`. Set `LORA_FREQ` to your region's frequency (e.g. 910.525 for USA/Canada, 867.5 for EU) before building.

save, then run:

```
pio run -e RAK_4631_Repeater
```

then you'll find `firmware.zip` in `.pio/build/RAK_4631_Repeater`

5.10. Q: Are there other MeshCore related open source projects?

A: [Liam Cottle](#)'s MeshCore web client and MeshCore Javascript library are open source under MIT license.

Web client: <https://github.com/liamcottle/meshcore-web>

Javascript: <https://github.com/liamcottle/meshcore.js>

5.11. Q: Does MeshCore support ATAK

A: ATAK is not currently on MeshCore's roadmap.

Meshcore would not be best suited to ATAK because MeshCore:

clients do not repeat and therefore you would need a network of repeaters in place

will not have a stable path where all clients are constantly moving between repeaters

MeshCore clients would need to reset path constantly and flood traffic across the network which could lead to lots of collisions with something as chatty as ATAK.

This could change in the future if MeshCore develops a client firmware that repeats.

[Source](#)

5.12. Q: How do I add a node to the [MeshCore Map](#)

A:

To add a BLE Companion radio, connect to the BLE Companion radio from the MeshCore smartphone app. In the app, tap the `3 dot` menu icon at the top right corner, then tap `Internet Map`. Tap the `3 dot` menu icon again and choose `Add me to the Map`

To add a Repeater or Room Server to the map, go to the Contact List, tap the `3 dot` next to the Repeater or Room Server you want to add to the Internet Map, tap `Share`, then tap `Upload to Internet Map`.

You can use the same companion (same public key) that you used to add your repeaters or room servers to remove them from the Internet Map.

5.13. Q: Can I use a Raspberry Pi to update a MeshCore radio?

A: Yes.

Below are the instructions to flash firmware onto a supported LoRa device using a Raspberry Pi over USB serial.

“ Instructions for nRF devices like RAK, T1000-E, T114 are immediately after the ESP instructions

For ESP-based devices (e.g. Heltec V3) you need:

- Download firmware file from <https://flasher.meshcore.io>
- Go to the web site on a browser, find the section that has the firmware up need
- Click the Download button, right click on the file you need, for example,
`Heltec_V3_companion_radio_ble-v1.7.1-165fb33.bin`
- Non-merged bin keeps the existing Bluetooth pairing database
- `Heltec_v3_companion_radio_usb-v1.7.1-165fb33-merged.bin`
- Merged bin overwrites everything including the bootloader, existing Bluetooth pairing database, but keeps configurations.
- Right click on the file name and copy the link and note it for later use here is an example:
`https://flasher.meshcore.io/releases/download/companion-v1.7.1/Heltec_v3_companion_radio_ble-v1.7.1-165fb33.bin`
- Run:
- `wget https://flasher.meshcore.io/releases/download/companion-v1.7.1/Heltec_v3_companion_radio_ble-v1.7.1-165fb33.bin` to download the firmware file for your device type. or the version you need - USB, BLE, Repeater, Room Server, merged bin or non-merged bin
- If the above wget command only downloads a very small file (10K bytes instead of more than 100K byte, use this command instead:
- `wget --user-agent="Mozilla/5.0" --content-disposition "https://flasher.meshcore.io/releases/download/companion-v1.7.1/Heltec_v3_companion_radio_usb-v1.7.1-165fb33.bin"`
- Confirm the `ttyXXX` device path on your Raspberry Pi:
- Go to `/dev` directory, run ls command to find confirm your device path
- They are usually `/dev/ttyUSB0` for ESP devices
- For ESP-based devices, install esptool from the shell:
- `pip install esptool --break-system-packages`
- To flash, use the following command:
- For non-merged bin:
- `esptool.py -p /dev/ttyUSB0 --chip esp32-s3 write_flash 0x10000 .bin`
- For merged bin:
- `esptool.py -p /dev/ttyUSB0 --chip esp32-s3 write_flash 0x00000 .bin`

Instructions for nRF devices:

For nRF devices (e.g. RAK, Heltec T114) you need the following:

- Download firmware file from <https://flasher.meshcore.io>
- Go to the web site on a browser, find the section that has the firmware up need
- You need the ZIP version for the adafruit flash tool (below)
- Click the Download button, right click on the ZIP file, for example:
`RAK_4631_companion_radio_ble-v1.7.1-165fb33.zip`
- Right click on the file name and copy the link and note it for later use here is an example:
`https://flasher.meshcore.io/releases/download/companion-v1.7.1/RAK_4631_companion_radio_ble-v1.7.1-165fb33.zip`
- Run:

- `wget https://flasher.meshcore.io/releases/download/companion-v1.7.1/RAK_4631_companion_radio_ble-v1.7.1-165fb33.zip` to download the firmware file for your device type. or the version you need - USB, BLE, Repeater, Room Server, ZIP file only
- Confirm the `ttyXXX` device path on your Raspberry Pi:
- Go to `/dev` directory, run `ls` command to find confirm your device path
- They are usually `/dev/ttyACM0` for nRF devices
- For nRF-based devices, install `adafruit-nrfutil`
- `pip install adafruit-nrfutil --break-system-packages`
- Use this command to flash the nRF device:
- `adafruit-nrfutil --verbose dfu serial --package RAK_4631_companion_radio_usb-v1.7.1-165fb33.zip -p /dev/ttyACM0 -b 115200 --singlebank --touch 1200`

To manage a repeater or room server connected to a Pi over USB serial using shell commands, you need to install `picocom`. To install `picocom`, run the following command:

- `sudo apt install picocom`

To start managing your USB serial-connected device using `picocom`, use the following command:

- `picocom -b 115200 /dev/ttyUSB0 --imap lfcrLf`

From here, reference repeater and room server command line commands on MeshCore github wiki here:

- <https://github.com/meshcore-dev/MeshCore/wiki/Repeater-&-Room-Server-CLI-Reference>

5.14. Q: Are there are projects built around MeshCore?

A: Yes, there are many. MeshCore's protocol is open source using the MIT license. The MIT license and the open source protocol makes it very easy for the MeshCore community to build new firmware for radios, applications on mobile devices, map tools, and analysis tools, and integration with other projects like Home Assistant.

As new MeshCore community projects become available on a weekly basis, we have stopped tracking them here in this FAQ. [samuk](https://github.com/samuk/awesome-meshcore/blob/main/README.md) maintains a very exhaustive list of MeshCore community project at <https://github.com/samuk/awesome-meshcore/blob/main/README.md>. [samuk](https://github.com/samuk/awesome-meshcore/blob/main/README.md) accepts PRs and merges them regularly.

5.15. Q: Are there client applications for Windows or Mac?

A: Yes, the same iOS and Android client is also available for Windows and Mac. You can find them together with the Android APK here:

<https://files.liamcottle.net/MeshCore>

Both the Windows and Mac versions of the client app are fully unlocked and are free to use.

5.16. Q: Are there any resources that compare MeshCore to other LoRa systems?

A: Here is a list of MeshCore comparison resources:

The Comms Channel on YouTube:

<https://www.youtube.com/watch?v=guDoKGs02Us>

MeshCore Advantages by MCarper:

<https://github.com/mikecarper/meshfirmware/blob/main/MeshCoreAdvantages.md>

[Meshcore vs Meshtastic](#) by austinmesh.org

<https://www.austinmesh.org/learn/meshcore-vs-meshtastic/>

Revision #5

Created 2026-05-03 05:05:09 UTC by Mesh America Admin

Updated 2026-06-08 19:55:25 UTC by Mesh America Admin