Mira version 2.6.0 Release Notes

Table of contents:

Features

Faster root restarts - New default behaviour

Mira 2.6.0 introduces a new way of starting a root node, intended for when the root node is being restarted. This mode can reduce network downtime from hours to seconds, without the need for other nodes to rejoin! In 2.6.0, a root will start up as a mesh node and listen for any running Mira networks. It will join the network if the credentials match as a regular mesh node would. After joining, however, it will turn on its root capabilities and start being the coordinator of the network.

In a root restart scenario, we rely on the mesh nodes still running the network for some time after the root powers down, and the root can rejoin the same network it just left. The root node attempts to find an existing network for two minutes, before giving up. In our testing the root was successful in rediscovering the network within the timeout over 99.9% of the time.

This means that the mesh nodes never have to lose sync, or go back to the scanning phase, and upstream communication is almost undisturbed by a root restart. Downstream communication will take longer to reestablish, however, due to RAM being flushed on the root in the restart. The root must re-learn the network topology in order to establish the downstream routes. This could take longer than a usual network startup due to the behaviour explained in: **Downstream links can take a long time to establish.**

We retained the previous startup behaviour of a root node but with a changed name. This is useful when starting a brand new network and you don't want to wait 2 minutes for the scanning to timeout. This introduces another network mode in `mira_net_mode_t`:

- "`MIRA_NET_MODE_ROOT_NO_RECONNECT`" - Same startup behaviour as previous versions.
- "`MIRA_NET_MODE_ROOT`" - Will scan as a mesh node for 2 minutes on startup.

Device monitoring APIs

Topology data from root node

`mira_diag_net_get_topology()` introduces functionality to get information about the current topology. Upon calling `mira_diag_net_get_topology()` the registered callback will be called for every node in the network with information about its respective parent. With this information it's possible to build a topology graph showing how packets are routed in the mesh network.

Neighbour information

`mira_diag_net_get_neighbour_info()` introduces functionality to get information about a node's immediate neighbours. Upon calling `mira_diag_net_get_neighbour_info()` the registered callback will be called for every neighbour that the node has discovered.

The API will supply the following information about every neighbour:

- Address
- Link metric
- How many radio exchanges the link metric value is based on
- RSSI
- Rate

Packet statistics

`mira_diag_mac_get_statistics()` introduces functionality to get statistics about packets being sent and received by a node. It includes how many packets have been sent, if they are unicast or multicast, and the same for received packets. It also tracks failed transmissions. It can be used to detect risk of data overload. See the documentation for more details.

Frontend controlled by the Nordic Semiconductor SoftDevice

Since version 2.5.1, the SoftDevice can control frontend modules configured by Mira. Mira leaves the frontend module in idle (the most power efficient state), and the SoftDevice sets it to 'bypass TX' when using the radio. This enables both concurrent BLE capabilities and low current consumption. Keep in mind that use of concurrent BLE will still affect the current consumption of the device.

API to check if a node is reachable

`mira_net_is_reachable()` adds the possibility for a root to check if it has established a route to a given node. This does not guarantee that the node is reachable, because the node could have gone offline without the root knowing it, but it is an indicator that the root has enough information about the given node to attempt transmission.

MiraMesh timer API

A low level timer interface has been added to allow MiraMesh users to save RTCs, by sharing the one used by MiraMesh. For users of FreeRTOS this means the task scheduler can be tickless without using its own RTC. This has been implemented in the MiraMesh and FreeRTOS example on our GitHub. See the documentation for more details.

rf_slots_* API additions

LQI

`rf_slots_get_packet_lqi()` has been added to get a link quality index for 802.15.4 packets sent by the rf_slots API.

CCA

`rf_slots_get_cca()` has been added to get CCA status for 802.15.4 packets sent by the rf_slots API if CCA has been turned on.

Slot priority

`mira_radio_timeslot_schedule_immediatly()` and `mira_radio_timeslot_schedule_at()` now takes a priority parameter. Use it to increase priority of timing critical radio slots.

Low frequency clock source configuration on nRF devices

Mira now supports the use of external low frequency clock sources instead of using a crystal. This enables the use of more accurate and temperature compensated clock generators. Configuration is done at compile time of `isr_vector.c` by preprocessor definitions. The configuration controls the hardware register `LFCLKSRC` of the nRF52840 and nRF52832 at startup. There are three modes available:

1. External Crystal (same as previous versions)
   - Default configuration
   - `LFCLKSRC->SRC = 1`
   - `LFCLKSRC->BYPASS = 0`
   - `LFCLKSRC->EXTERNAL = 0`
2. External Source (low swing)
   - Select by defining `MIRA_LFCLK_EXTERNAL_SOURCE_BYPASS_DISABLED`
   - `LFCLKSRC->SRC = 1`
   - `LFCLKSRC->BYPASS = 0`
   - `LFCLKSRC->EXTERNAL = 1`
3. External Source (rail-to-rail swing)
   - Select by defining `MIRA_LFCLK_EXTERNAL_SOURCE_BYPASS_ENABLED`
   - `LFCLKSRC->SRC = 1`
   - `LFCLKSRC->BYPASS = 1`
   - `LFCLKSRC->EXTERNAL = 1`

Improvements

Updated to nRF5 SDK 17.0.2 and SoftDevice 7.2

Mira now uses nRF5 SDK version 17.0.2 and SoftDevice S132/S140 7.2. SoftDevice S140 uses 4KiB more flash memory. Both SoftDevices use slightly more RAM. Custom link scripts will have to be updated because of this (see libmira's `platform/$TARGET/nrf52_xxaa_s132.ld` or `platform/$TARGET/nrf52840_xxaa_s140.ld`). This might affect upgrades if the SoftDevice and the rest of the firmware are updated separately on the target. In that case a workaround could be to have an intermediate application that has a copy of the ISR vector at both starting addresses. For more information see Annex A.

Faster Flash Access on nRF devices

Flash erase and write times have been significantly reduced on nRF devices. The architecture requires the operations to be 4-byte aligned in order to enable fast operation. Therefore it is strongly recommended that all flash operations are 4-byte aligned.

Bugfixes

Unroutable nodes

Resolved an issue where nodes and their children could become unroutable following a network startup/restart. This meant that downstream links would not establish, but upstream links worked. The nodes affected (if any) were random at each startup. The reason for this was a race condition between different packets, which has been fixed in version 2.6.0.

Poor NFC range

Resolved an issue where some nRF52840 chips experienced very poor NFC range. This was due to a bug in the Nordic Semiconductor SoftDevice. It has been fixed in the latest SoftDevice version, which is included in Mira version 2.6.0.

Changes

SWI1 on nRF52

Since 2.5.1, Mira on nRF52 uses SWI1's IRQ.

Projects using MiraOS with a custom isr-vector-file need to point the entry for `SWI1_EGU1_IRQHandler` to `mira_SWI1_EGU1_IRQHandler`.

Projects using MiraMesh need to add miramesh_swi1_irq_handler to the isr-vector or add a `SWI1_EGU1_IRQHandler` function that calls `miramesh_swi1_irq_handle`r.

Memory allocated at startup increased

We have moved some buffers around from being statically allocated at compile time to being dynamically allocated at network startup. This means that `MIRA_MEM_SET_BUFFER()` now requires an extra 2 592 bytes. This does not, however, mean that the total RAM usage increases from previous versions.

Limitations

Downstream links can take a long time to establish

For downstream (root to node) links to work, the root must have received a special package with the routing information to that particular node. This package is sent from the node on three events:

1. When the node joins the network
2. When the node changes parent
3. On a 1 hour interval

If the 1st package gets lost, and the network is very stable with no parent changes, it would take an hour for the link to establish. The reason for the long interval is to reduce network metadata overhead.

Root reboot scenario

With the new root reboot behaviour, described in Faster root restarts - New default behaviour, mesh and leaf nodes do not have the need to rejoin the network. Fully rebuilding all downstream links after a root reboot will then depend on parent changes or the hour long interval, making the process take up to one hour. This behaviour will be improved in coming versions of Mira.

Mira startup crash with running radio protocols - nRF

Mira can fail to start (resulting in a watchdog reset) if a radio protocol is running and using the radio at Mira initialization. This affects Mira versions 2.5.1 and up, on nRF builds using the Nordic Semiconductor SoftDevice. A normal use case where this can occur is if one uses a bootloader that runs a radio protocol, such as BLE, and then starts Mira without disabling the radio protocol first. Therefore it is **strongly recommended** to disable any and all radio protocols before starting MiraOS or calling `mira_net_init()` in MiraMesh.

Upcoming changes

Deprecation of C89:

Support for C89 is deprecated and will be dropped in the next minor release.

Annex A - Upgrade to the new SoftDevice

2.6.0 uses newer SoftDevice versions that consume some more RAM and on the nRF52840 platform, it also uses more flash. This may cause problems when upgrading the firmware via FOTA if the SoftDevice and the application are updated separately.

The SoftDevice S140 version 6.1 wants the application's isr-vector to start at address `0x26000` and SoftDevice S140 version 7.2 wants it to start at address `0x27000`.

The best way to handle this is to update the SoftDevice and application firmware at the same time. If that is not possible another possibility is to upgrade to an intermediate version of the firmware that can be used with both SoftDevice versions.

Build a firmware compatible with both softdevice versions

Change the old Mira version's `platform/nrf52840ble-net/nrf52840_xxaa_s140.ld` file's FLASH, SOFTDEVRAM and RAM entries to be the same as the one from 2.6.0. Then rebuild the firmware with this new linker-script and the old libmira.

That firmware will now work with the new softdevice, but not the old one. To also make it work with the old one, copy the flash page from `0x27000` to `0x26000` in the resulting file. Given a hex file that does not contain the SoftDevice image, this can be done with:

```
arm-none-eabi-objcopy -O binary fw.hex tmp.bin
dd bs=4096 count=1 if=tmp.bin of=isr.bin
cat isr.bin tmp.bin > isr+fw.bin
arm-none-eabi-objcopy -I binary --change-addresses 0x26000 -O ihex  isr+fw.bin
isr+fw.hex
rm tmp.bin isr.bin isr+fw.bin
```

Then upgrade the nodes with the updated firmware, then the new softdevice and finally the firmware built with 2.6.0.

If you need help in upgrading your application to 2.6.0, please contact LumenRadio support.