Mira version 2.8.1 Release Notes

Table of Contents

2.8.1

Bugfixes

Back-to-back exiting processes

In MiraOS 2.8.0, a bug was introduced that caused a crash when two processes exited consecutively. This issue has been resolved in 2.8.1.

Please note that due to this bug, **MiraOS 2.8.0** is now considered **obsolete**.

2.8.0

Bugfixes

Mix & Match different rates in a network

If the rates among nodes had big differences, the network could become unstable and ETX between nodes could be high. The previous workaround was to intentionally slow down more capable nodes to run slower rates and keep the difference small. It is now possible to use different rates without such concerns.

Note: when upgrading to 2.8.0 in order to utilize faster rates, it is best to first complete the Mira protocol version upgrade to 2.8.0 before changing rates. If not done in this two-step process an older Mira version may be used with unsupported differences in rate.

Different clock drift of high frequency and low frequency clocks

If the clock drift between the HF and LF clocks were different enough, sometimes nodes would join the network, but not receive any packets for a few minutes. Mira will now handle such clock drifts correctly.

Risk of long times for downstream route establishment during network startup

During startup of a network, information from all nodes is gathered at the root in order for it to establish downstream routes. This information could get lost for individual nodes due to packet loss, and was only present on slow intervals. This introduced risk of individual nodes taking a long time to be routable from the root. This affected all previous Mira versions. In 2.8.0 we have increased robustness in the gathering of this information and downstream routes should form faster and more reliably.

License dongle counter printout incorrect

In previous versions, when generating a second license for the same device id the counter got wrongfully incremented. Generating a third license would, correctly, not increase the counter. This behavior was only present in the printout, and not in the actual license counter. This is now fixed, and the printout correctly represents the license counter.

Flash operations during concurrent BLE could hang

When doing flash operations and running the BLE stack through the Softdevice, the flash API could end up in an unrecoverable state. This was due to the Softdevice canceling flash operations, in favor of BLE operations, and was not properly handled by the API, making `flash_is_busy()` never return false. This is now fixed and the `flash_is_busy()` will now return false and the operation should return the appropriate error when interrupted by the softdevice.

Neighbor table overflow handling

In previous versions, the stack didn't allow for having more nodes within range than was specified in the `max_nbrs` member in the `mira_net_config_t` struct. This was due to the stack being unable to prioritize between neighbors, and unexpected behavior could occur if internal lists overflowed. This is now fixed by having a prioritized list with clean up functions. This enables `max_nbrs` to be lower than the maximum number of nodes any given node will have within range. However, keep in mind that `max_nbrs` should be at least as high as the number of maintained links a node should be able to have.

Slow network startup after failed root rejoin

When running `MIRA_NET_MODE_ROOT` in versions 2.6.0 and up the root will try to rejoin the network upon startup/restart. It will timeout after 2 minutes and start normally. If this happens a bug caused the root to not send messages for another 5 minutes, prolonging the network rebuild process with 5 minutes. This is now fixed.

Sporadic slow license validation

Upon startup Mira validates the license. This was randomly much slower than usual. The problem went away when connecting a debugger. This was related to `sd_ecb_block_encrypt` calls and a bug in the Softdevice. A workaround is implemented, and the issue is fixed.

`mira_config_write()` failed if called when busy

When calling `cpu_userconfig_write()` while another write was in progress, the ongoing write could unpredictably fail. This is now fixed.

`mira_config_read()` didn't default to `UICR->CUSTOMER`

To enable backwards compatibility with older versions (when the `mira_config` was located in the `UICR` area) there is a default behaviour to read the UICR if a config in the now normal place is missing. In 2.6.0 and up this wasn't the case, and the function instead returned `MIRA_ERROR_NOT_INITIALIZED`. This is now fixed, and the function reads the `UICR` area instead.

BUSB-UART was not optimised by the linker

This meant that the module consumed RAM and FLASH resources despite being used or not. This is now fixed. nRF52840 builds that don't use the USB-UART driver can expect lower RAM/FLASH usage.

Sporadic low BLE output power with frontends

Due to an unforseen dependency in an interrupt routine, Softdevice control of a frontend could be inverted if the interrupt was called an uneven number of times. This does not normally happen, but can if radio slots fail. This leads to the frontend module being in an incorrect state, resulting in very low output power. This is now fixed.

Features

Configurable amount of FOTA clients served

A new parameter, `max_fota_clients`, has been added to the `mira_net_config_t` struct, and the Border Gateway configuration file (`MIRA_GW_MAX_FOTA_CLIENTS`). It specifies how many child nodes the given node can send its FOTA image to simultaneously. This can drastically shorten FOTA transfer times in networks, particularly in network with differences in net rate. If left unspecified, the old default value 2 will be used. See annex A for a more detailed performance evaluation.

`max_fota_clients` needs to be lower than `tx_queue_size` - 6. Serving many clients, particularly clients with slower net rate than yourself, will result in many outgoing packets waiting in the queue. Therefore one may need to also increase `tx_queue_size` (available in the same struct) when increasing `max_fota_clients.c`

Increased TX queue size for border gateway

The TX queue size for the border gateway host in increased to 64, from the previous value 16. This is to allow for serving more FOTA clients without overflowing the queue. The TX queue size for the gateway is not configurable by the user.

FOTA transfer client saves progress

In previous versions, if the FOTA transfer process between a child and a parent got interrupted, for whatever reason, the child started again from scratch. In 2.8.0 it continues where it left of. It still verifies the entire image with a checksum after a completed transfer. So if the transfer got aborted due to something that also created corrupt data it will pick up where it left of and fail the final verification and start over.

`mira_diag_get_rank()`

A diagnostics API extension that returns the RPL rank of the node. The rank is an arbitrary value for parent fitness that nodes use to evaluate which parent is best, where lower is better. The evaluation is not based on the parent rank alone, but the parent rank plus a penalty based on the quality and performance of the link to that parent. Keep in mind that rank is a relative value and is not comparable between different networks.

`mira_uart_deinit()`

This function de-initializes the UART interface. This is helpful when wanting to sporadically use UART without having the current consumption impact of leaving it initialized all the time.

Updates and improvements

More RAM used per entry in neighbor table

The neighbor table entry has grown by 8 bytes. This means that a firmware, build on 2.8.0, using the default neighbor table size of 64 entries will use 512 bytes more ram than in previous versions. However, because of better neighbor table overflow handling the neighbor table size can safely be significantly reduced.

Watchdog reset if not enough RAM is allocated

If not enough memory is allocated with the `MIRA_MEM_SET_BUFFER()` macro before running `mira_net_init()` it causes a watchdog reset. In previous versions execution silently halted.

Gateway scripts bourne shell compatible

The gateway scripts are now bourne shell compatible.

Windows support for mira_license.py

The licensing script, mira_license.py, previously didn't have support for Windows. This is now solved. The supported versions of dependencies are:

- nrfjprog: 10.22.0 external
- JLinkARM.dll: 7.88i
- pynrfjprog: 10.15.2

Reworked keep alive message mechanism

The Mira stack implements keep alive messages. They are messages that the stack generates when there a no other messages that the stack can use to sync its crystals. They are usually sent to the parent, which will answer with an acknowledgement, feeding the child with a good timing reference. These messages are generated after a few minutes if no other syncable message is heard by the node.

One problem with this approach can occur in large, silent networks with high fan in. If a parent has very many children to whom it also doesn't talk to very often, the children can start to collectively generate very many keep alive messages. Since the response is in the acknowledgement of the child-generated message, the children each need a response to their particular message and can't listen on an acknowledgement generated by another child. Ironically, this can, in this particular scenario, put a heavy load on the parent.

In 2.8.0, we have reworked certain beacon messages to be on a predictable schedule. This enables mesh and leaf nodes to listen for a beacon message (that would be sent by the stack anyway) and get sync that way, instead of generating unnecessary traffic. If this fails for some reason (like having a parent without this predictable schedule), the node will default back to sending keep alive messages.

Known issues

USB radio stick bootloader doesn't enter DFU mode after update

When starting the gateway with an older radio stick using USB it may need to update the USB DFU bootloader. The update will complete, but it will not enter the DFU mode. It needs to be in DFU mode to be able to also load the gateway radio firmware, so this hangs the gateway host program. A workaround is to manually set the USB radio stick into DFU mode by running the scrip `reset_to_dfu.py`, located in the tools folder.

Using gcc option -B results in infinite loop

When compiling an application with libmira.a using gcc option -B, compilation gets stuck in an infinite loop, compiling the application over and over again.

`mira_flash_write()` hangs if writing to page 0

Instead of throwing an error, the API hangs indefinitely if one tries to write to the invalid block 0.

Annex A – FOTA performance characterization

To understand how the new parameter, `max_fota_clients`, affects FOTA transfer speeds we have done some characterising tests. The tests have mainly been performed in simulation and are focused on one “step” in the FOTA transfer process, one server sending to multiple clients. This process is repeated many times in an actual FOTA transfer to a whole network, but the total transfer time is then dependent on the network topology. To keep it simple we isolated testing to just one transfer.

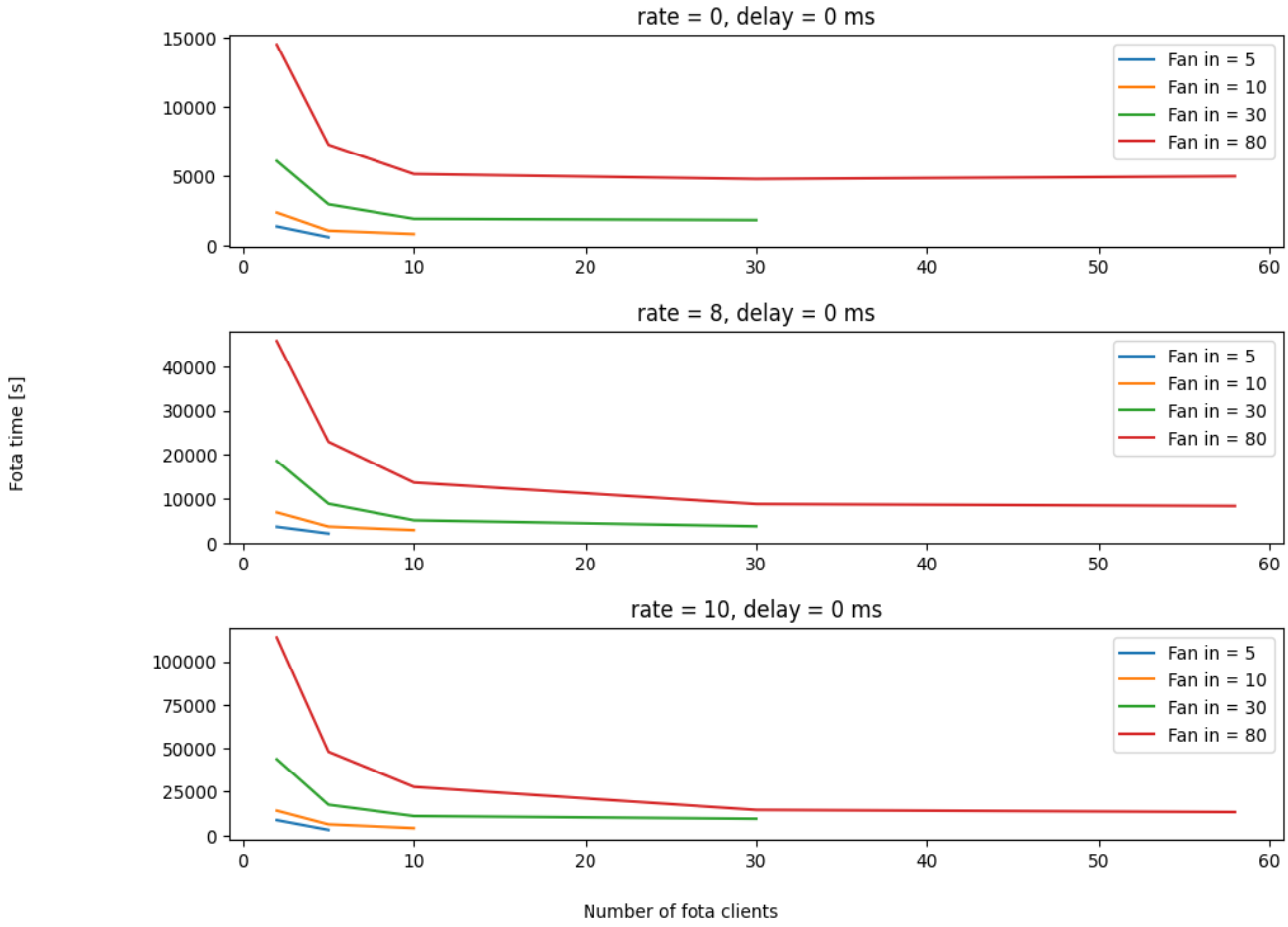
Test setup

All FOTA client nodes were placed in a uniform fashion on a circle with a rate 0 FOTA server in the middle of the circle. Fota transfer time was measured as a function of:

- number of clients in the test ("fan in” in plots): [5, 10, 30, 80]
- the rates of the clients: [0, 8, 10]
- number of clients served simultanously (`max_fota_clients`): [2, 5, 10, 30, 40, 58]
- tx queue size on the server: [16, 64]
- firmware size in bytes: 300*1024

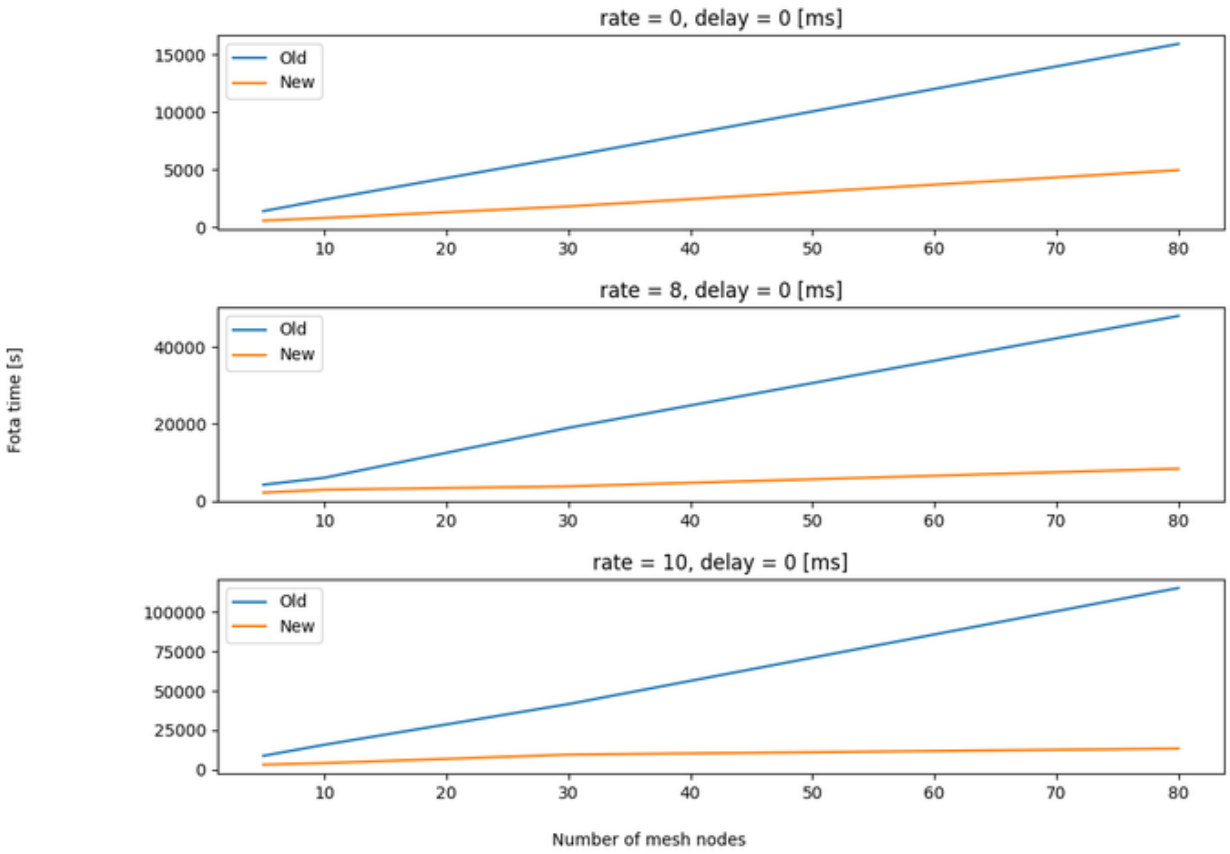
Results

Below is a plot of total time of serving all clientes (y-axis) vs number of clients served simultanously. It uses a 300 kB firmware and a TX queue size of 64. There are three plots with different client rates. The server runs rate 0 in all cases.



We see a clear improvement in all cases. The transfer time is halved (or more) by going to 5 clients served in all cases, with even lower transfer times for more clients served. The three plots does reach diminishing returns eventually, but at different stages. The client rate 0 reaches diminishing returns between 10 and 30 clients served, while the others reaches it after 30 clients served. There is no observed downside to having more clients served than needed, except for RAM usage.

To measure the difference in overall network update speed we set up a test with 80 nodes and measured the time it took for them to receive the firmware. Comparing the new highest tested `max_fota_clients`, 58, with the old default, 2, get the following plot of transfer time vs. nodes updated:



Here we see again that all cases benefit greatly from a higher `max_fota_clients`, but cases using slower rate on clients compared to servers benefit the most.