Over-the-Air Firmware Update in LoRaWAN Networks: A New Module-based Approach

Huy Dat Nguyen

PhD Student
at INZU team, IRISA

LPWAN days, 8-9 July 2024

Outline

1. Context and motivation

Motivation

Use case: Update 120KBytes of firmware in LoRaWAN BW125KHz, SF8
LoRa device Class C, EU868, duty cycle 10% → Update time: **1h01m**

↓

Idea: - To separate firmware to several modules
- Only modified modules need to be updated

Over-the-Air Firmware Update in LoRaWAN Networks:
A New Module-based Approach

↓

Aims to improve the performance of update process for LoRa-based devices in LoRaWAN networks:

- Update size
- Update time
- Energy consumption
- Network efficiency

2. State of the art

Middleware supports firmware update in IoT devices:

- Femto-containers: secure deployment, execution and isolation of a tiny virtual machine on low-power IoT devices
- End-to-End Mechanized Proof of an eBPF Virtual Machine for Micro-controllers

Drawbacks

- Less lightweight: programming scope limited
- More complex: a verification process required

Research works of modular programming and dynamic linking techniques for IoT devices:

- Contiki: static symbol table
- Dynamic TinyOS: dynamic symbol table

Drawbacks

Update restricted to application level

- GITAR: supports dynamic application and network level upgrades
- SOS: position independent code

Drawbacks

Do not couple modular programming and dynamic linking techniques with an OTA update process

2. State of the art

Research works investigating the usage of LoRa in Firmware update Over the Air are very limited:

- FUOTA specifications in LoRaWAN by LoRa Alliance
- Block-chain
- Multiple gateways approach
- Adaptive data rate
- …

Full image update

Drawbacks

- At least 3 partitions in flash required
- Huge energy consumption, update time
- Reboot required

- FLoRa

Incremental update

Drawbacks

- At least 4 partitions in flash required
- Reboot required

3. Dynamic module-based FUOTA method in LoRaWAN

Advantages of module-based architecture:

- Easy to add, delete, modify, replace and maintain modules
- Compared to container-based approach: more lightweight, less complex to implement

Advantages of dynamic module-based FUOTA method:

- Only modified sections need to be updated instead of entire firmware
- Compared to full-image replacement: less memory and energy consumption, more network efficiency, reboot not required

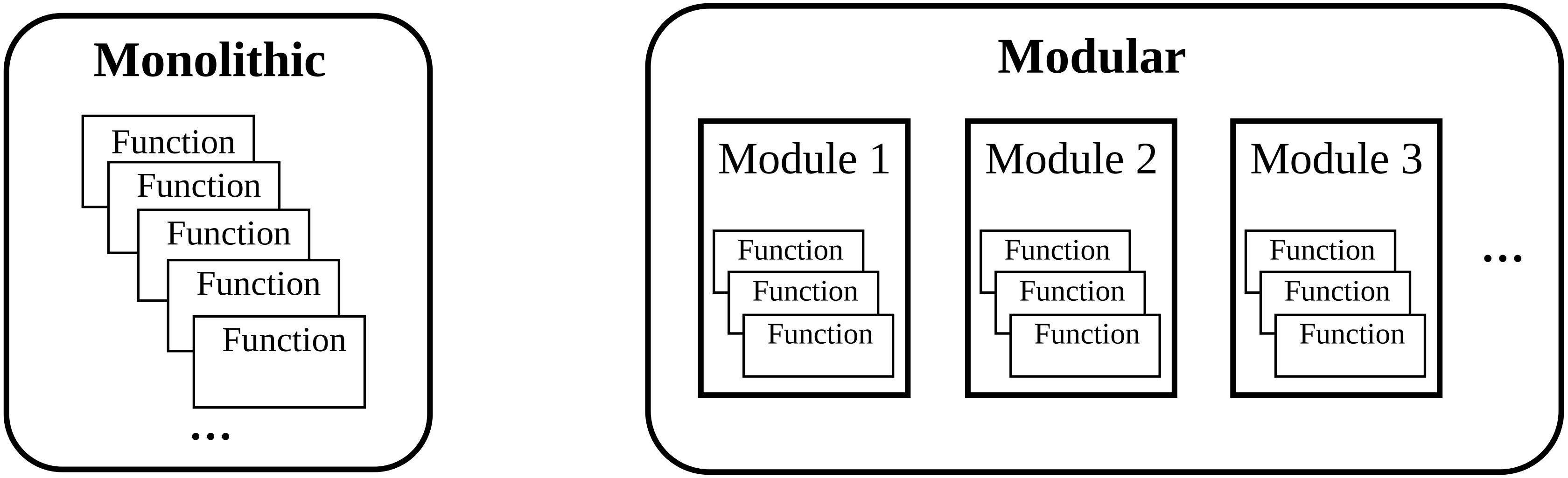
Based on 2 pillars:

2.1. Module-based approach: the monolithic firmware is replaced by a module-based software controlled by a micro-system

2.2. FUOTA method: takes benefit from the splitting of the firmware into modules, and has been implemented in an usual LoRaWAN architecture

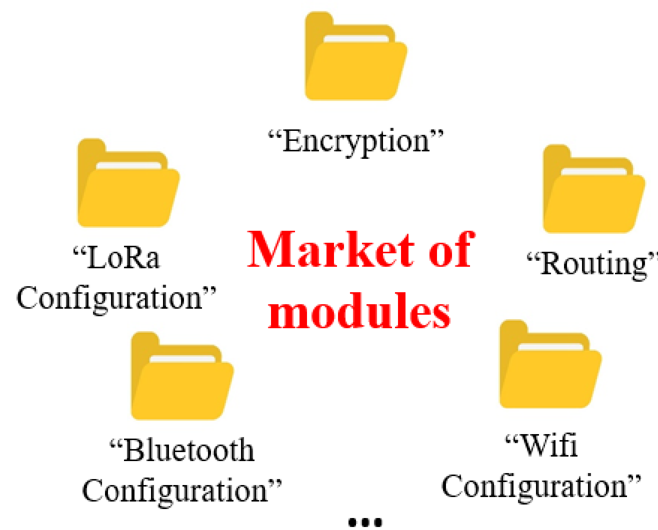
3. Dynamic module-based FUOTA method in LoRaWAN

3.1. Module-based approach

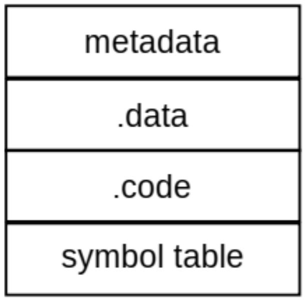


Advantages of modular design:

- Functionalities are grouped into modules
- The same module can be replicated to several devices
- Module can be encapsulated and released into the market so that other systems can reuse it

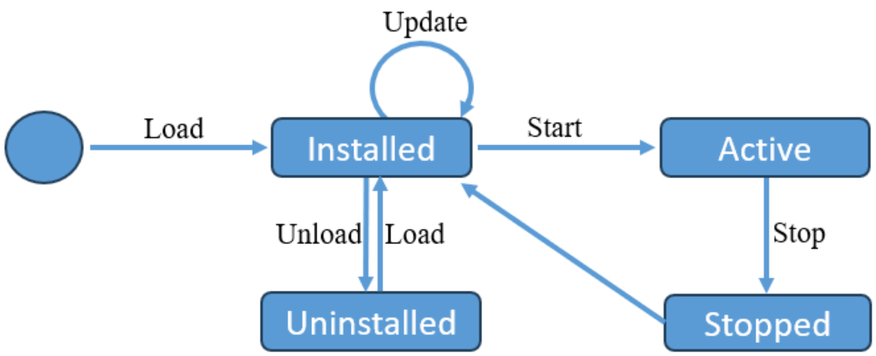


Loadable module
- Be independently created, modified, replaced, or exchanged
- Contains not only code and data, but also names of referenced symbols that are dynamically loaded.
- Be stored in flash memory
- Be reusable and managed in a systematic manner



Module composition

Life cycle:
- Micro-system verifies if the module is active (being used by others) before updating
- Dependent modules are also notified to use the update



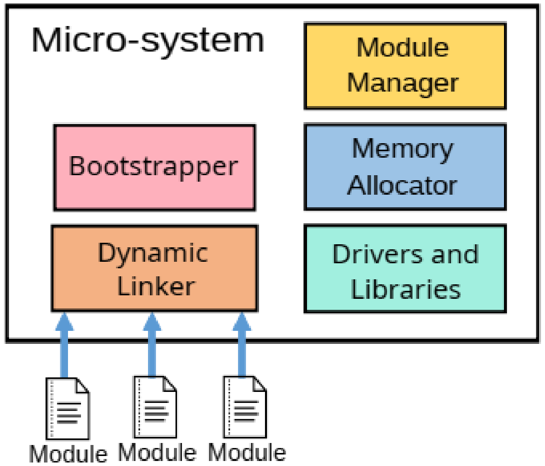
Life cycle of module

Property:
- Metadata: name, version, checksum
- Be used for verification and be removed before the module is loaded

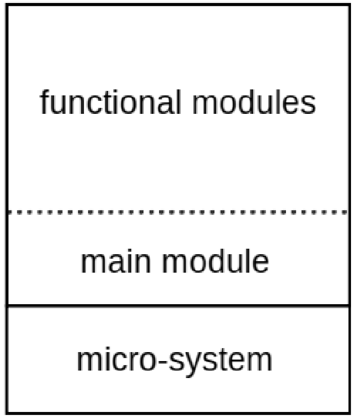
Side effect:
- Wrong data format
- Solution: delete data section when unloading module

3. Dynamic module-based FUOTA method in LoRaWAN

3.1. Module-based approach



General structure

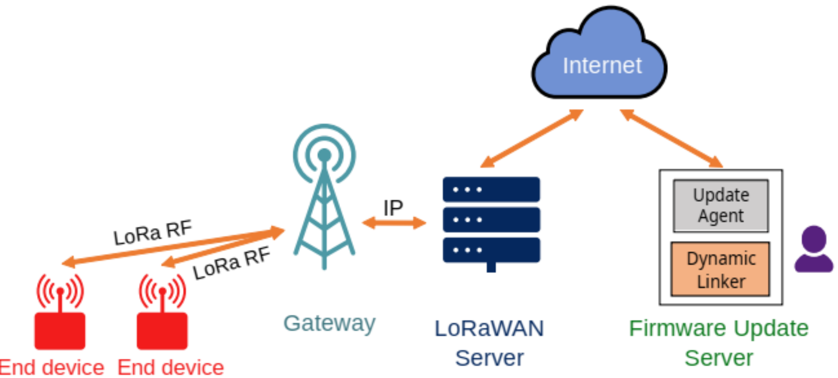


Memory layout

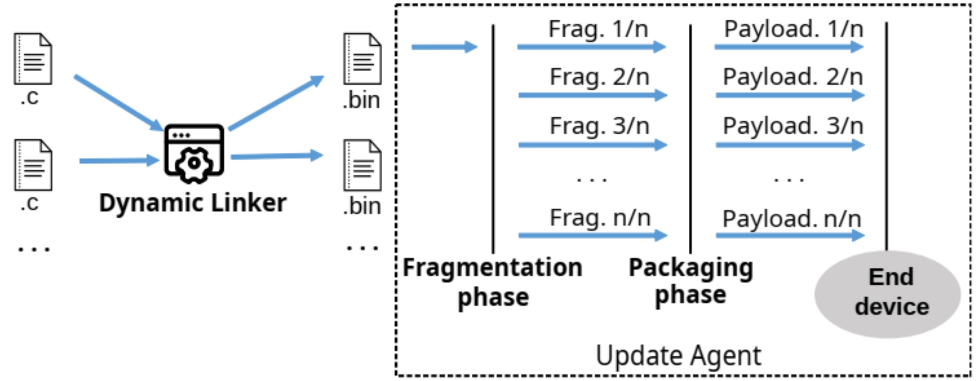
- Micro-system is installed directly on bare metal
- DL enables modules to be loaded at run-time in two different ways: in flash or in RAM
- MM controls the operation on modules
- MA optimizes the memory usage

3. Dynamic module-based FUOTA method in LoRaWAN

3.2. FUOTA method

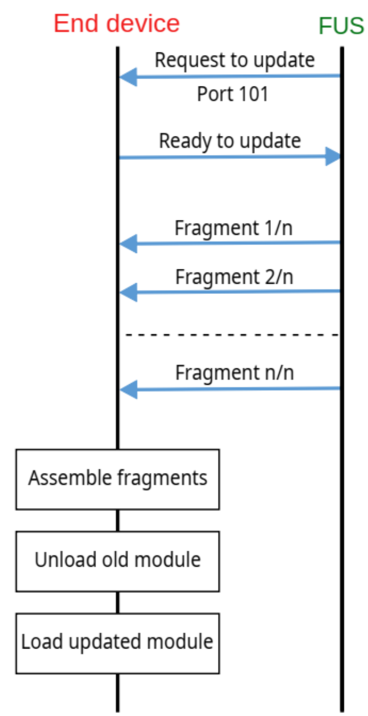


FUOTA architecture



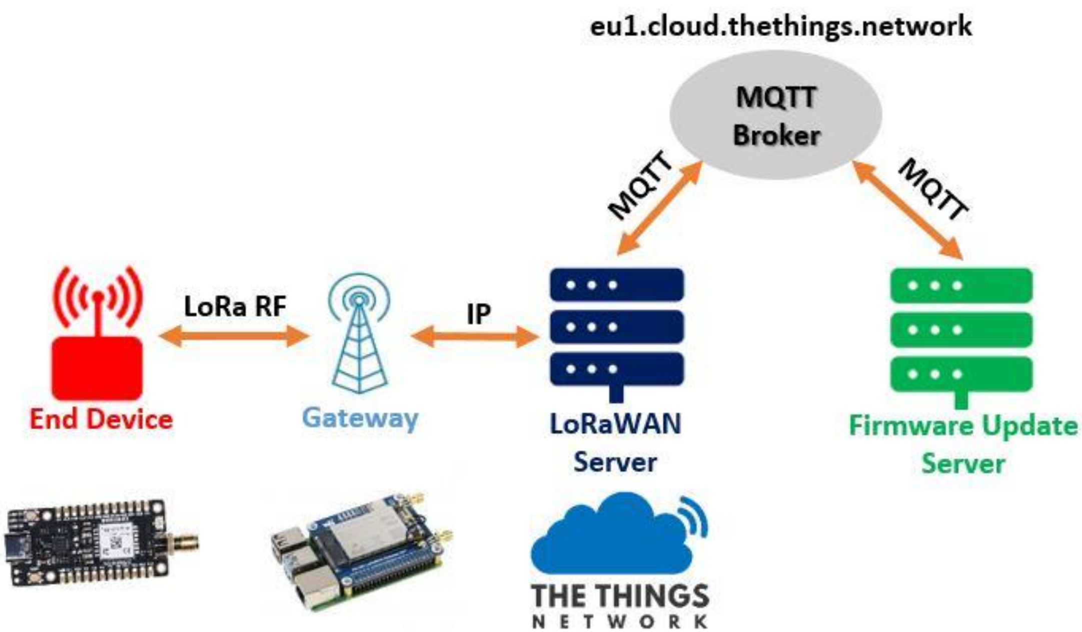
Firmware update server (FUS)

- *Dynamic update*
- *Low flash memory footprint*
- *Size and network efficiency*



FUOTA working principle

3.3. Implementation



STM32WLE5JC end-device

Sx1302 868MHz LoRaWAN Gateway

Flash memory address	Size	Name
0x0803 F800 - 0x0803 FFFF	2 kbytes	Page 127
0x0803 F000 - 0x0803 F7FF	2 kbytes	Page 126
...	...	...
0x0801 9000 - 0x0801 97FF	2 kbytes	Page 50
0x0801 8800 - 0x0801 8FFF	2 kbytes	Page 49
...	...	...
0x0800 0800 - 0x0800 0FFF	2 kbytes	Page 1
0x0800 0000 - 0x0800 07FF	2 kbytes	Page 0

Modules (Page 127 – Page 50)

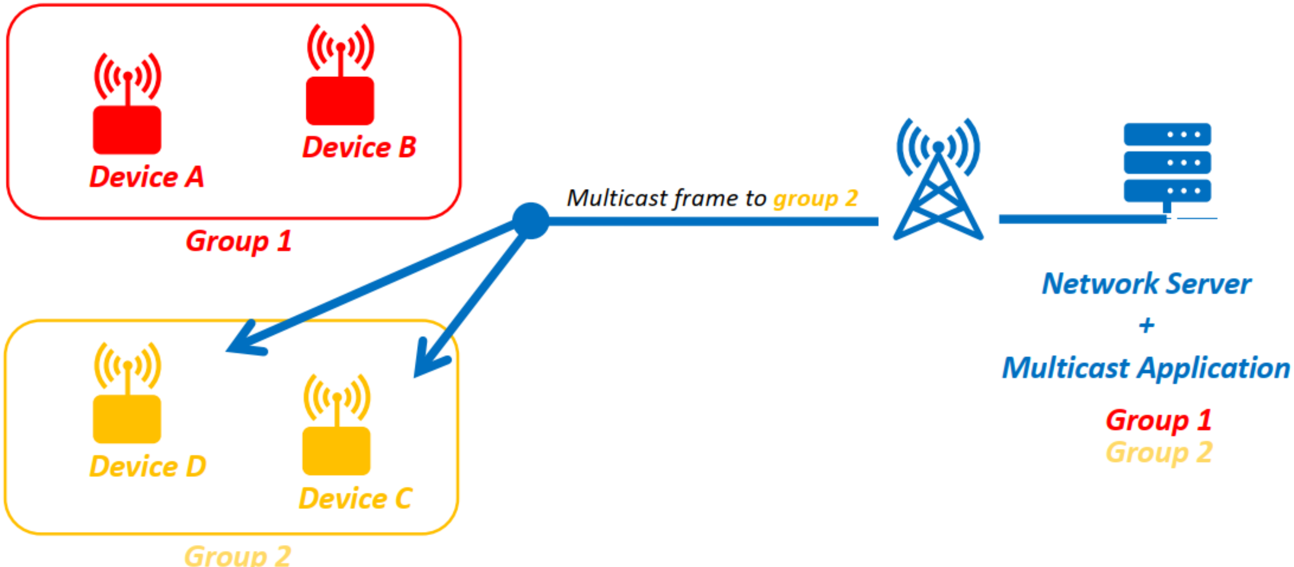
Micro system (Page 49 – Page 0)

Flash memory of the STM32WLE5JC MCU

3. Dynamic module-based FUOTA method in LoRaWAN

3.4. Multicast approach

- Multicast means sending a single downlink frame to several end-devices at the same time
- End-device needs to be part of a multicast group that the server wants to send a unique frame for
- Group definition can be done before deployment or remotely after deployment (add/delete/modify)



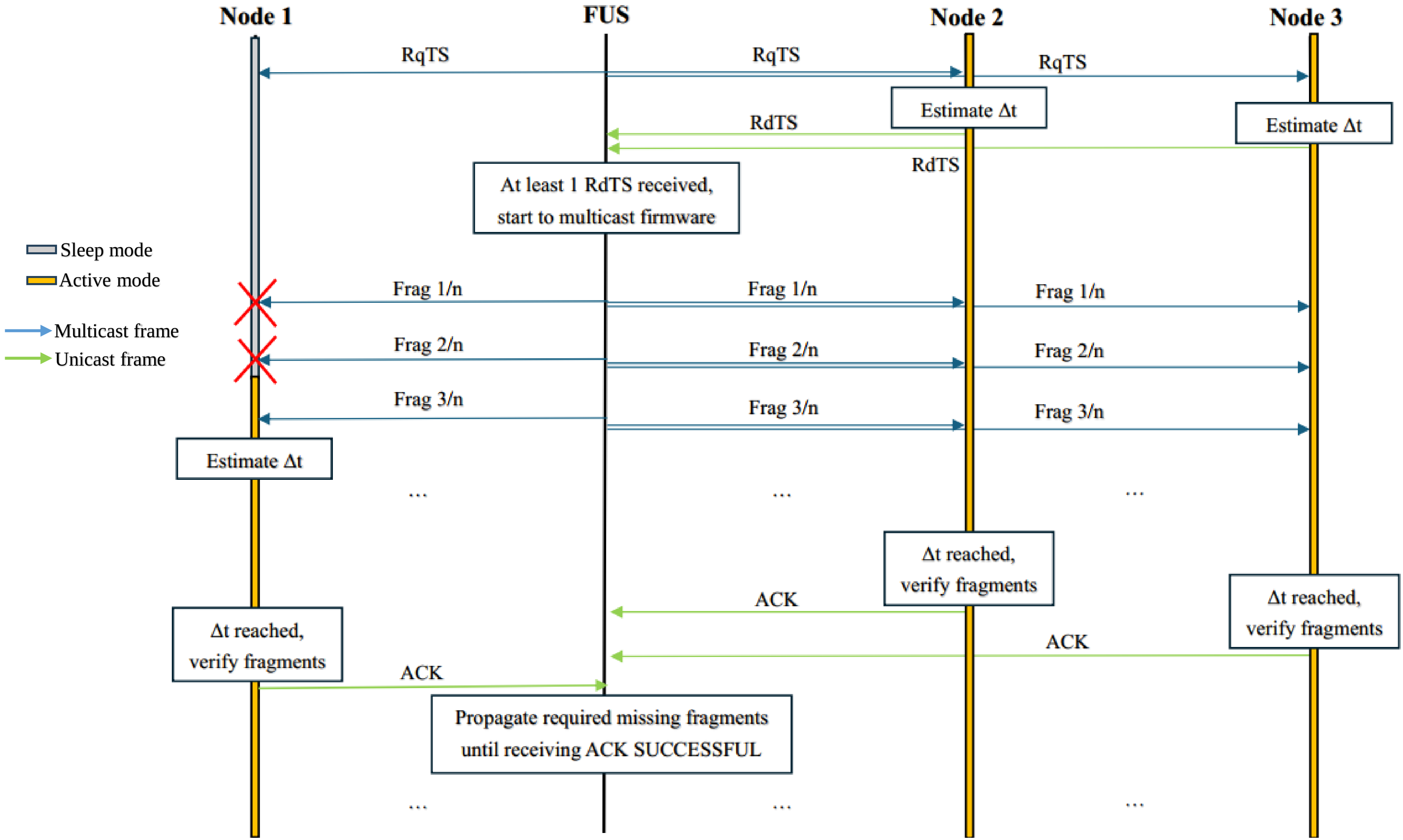
Multicast implementation*

Advantages:

- When a large number of devices performs the same behaviors, firmware update should be propagated to a specific set of devices
- Easy to scale FUOTA application

**Image source: Book LoRaWAN Advanced*

3.4. Multicast approach



4. Experimental results

Objective: to compare the approach with existing monolithic architecture in terms of:

- Memory consumption
- Update size and network load

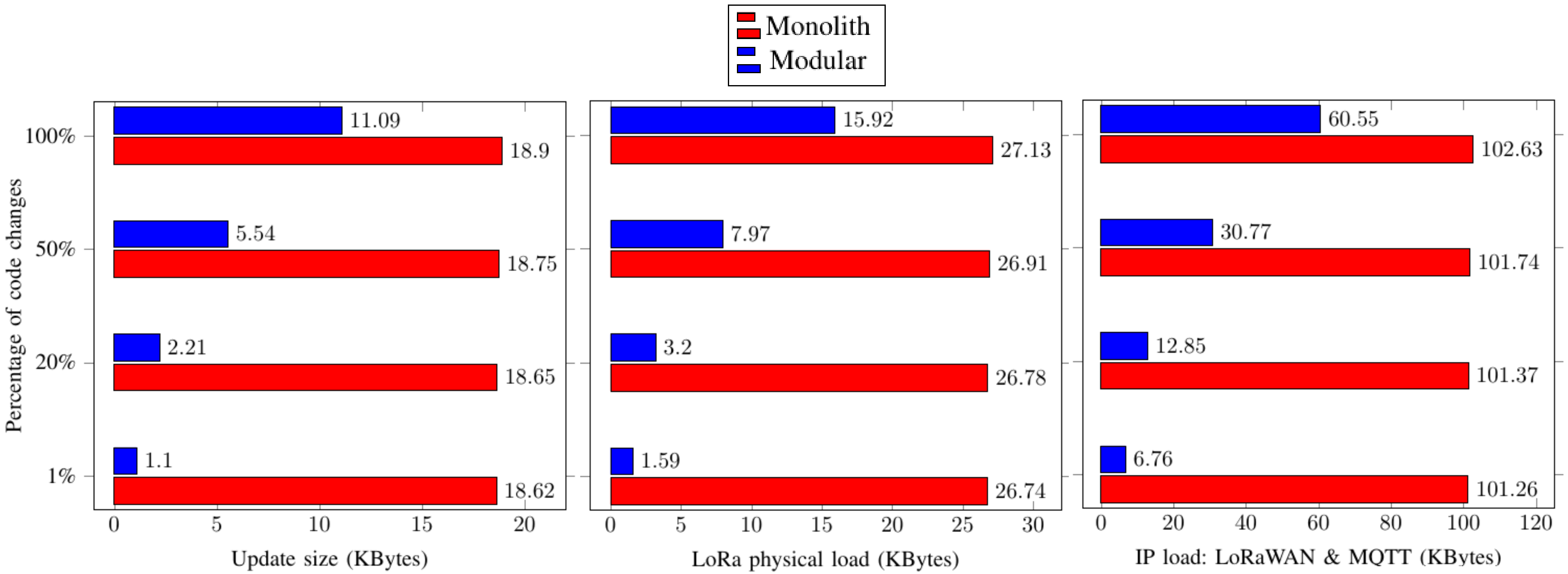
Parameters	Value
LoRa frequency band	868 MHz
LoRa spreading factor	7
LoRa bandwidth	125 kHz
LoRa coding rate	4/5
Fragment size	200 bytes

- Monolithic: is composed of ten function blocks (encryption, wireless configuration, sensor data collection, etc).
- Module-based: a set of 10 modules whose size varies from 0.71 to 1.36 kB
- Various modifications were randomly applied on this code during our experiments.

	Section	Flash required (kB)	RAM required (kB)
	Bootloader	17.96	2.16
Monolithic	Active firmware	18.6	1.54
	Downloaded firmware	20	0
	Total	*56.56*	*3.7*
	Micro-system	14.46	2.13
Modular	Modules	11.27	2.78
	Total	*25.73*	*4.91*

Memory consumption between monolith and modular design

4. Experimental results



- Modular denotes a higher performance in terms of the update size with a gain ratio up to 17, with 1% of code changes.
- Obtaining a small load in the LoRa network is particularly beneficial: it passes from 27 kB for the monolithic approach to 1.6 kB when using dynamic modules, with 1% of code changes.

5. Conclusion and future works

Contributions

- Module-based architecture for IoT devices’ firmware
- Dynamic FUOTA method
- Memory and network efficiency
- Experimental results obtained on a small testbed show that the solution we propose optimizes the update size and the network traffic up to 17 times compared to the traditional monolithic-based method

Future works

- Experiments to evaluate the update time and energy efficiency in multicast context
- Feasibility of a dynamic modular FUOTA in opportunistic networks

THANKS FOR YOUR ATTENTION!