LoRaWAN Firmware Update Over-The-Air (FUOTA)

Julien Catalano

Principal Architect – Head of Standards at Kerlink, France
LoRa Alliance Technical Committee FUOTA Working Group Chair, Fremont, CA, United States
E-mail: j.catalano@kerlink.fr

Received 30 December 2019; Accepted 21 June 2020;
Publication 30 April 2021

Abstract

Firmware Update is a key feature for IoT, especially for LPWA end-devices with 10+ years of lifetime. LoRaWAN Firmware Update Over-The-Air is a set of application layer specifications, including Multicast Setup, Fragmentation, Clock Synchronization, Firmware Management as well as Multi-Package Access, enabling the delivery and management services of firmware updates to several end-devices.

Keywords: Firmware update, OTA, multicast, fragmentation, forward error correction, clock synchronization, firmware management.

1 Introduction

Low-Power Wide Area (LPWA) Internet of Things technologies allow service providers and enterprises to offer long-range connectivity for battery-powered objects that consume little energy. Some popular use cases for LPWA devices include temperature and environmental sensors; smart meters for gas, electricity and water; asset and inventory tracking; agribusiness support; and industrial monitoring. One of the common attributes of LPWA devices is device longevity: generally, an LPWA-equipped sensor or a meter is expected to operate for more than 10 years without battery replacement or charging. But low-power and longevity requirements of Internet of Things (IoT) end-devices make LPWA firmware update an essential and challenging component of an end-to-end IoT solution. During the lifetime of the device, many things can change. Objective of the device may evolve, regulation and standards may change, vulnerabilities, security issues or bugs may be found.

LoRa Alliance® Technical Committee has created a dedicated working group to tackle the design of the specifications which bring the building blocks of the firmware update service for LoRaWAN end-devices: the Firmware Update Over-The-Air (FUOTA) Working Group. The group took the challenge to create generic protocols to address the following:

- Synchronize the real-time clock of LoRaWAN end-devices
- Create and manage fragmentation sessions to send a large block of data to LoRaWAN end-devices, with specific Forward Error Correction (FEC) code
- Schedule and manage temporary multicast sessions on LoRaWAN end-devices, with either continuous (Class C) or slotted (Class B) receptions
- Operate basic firmware management operations, query of versions and reboot the end-device
- Discover aforementioned packages and others, send multiple commands in a single payload to limit the number of message exchanges.

All those protocols have been designed to be usable on any existing or future LoRaWAN link layer specification versions, including current 1.0.x and 1.1.x, without modifications. The design requirement imposed construction of these protocols at the application layer.

A typical firmware update campaign goes as follow. The network application selects a group of end-devices to be updated, typically based on their hardware and current firmware versions. The Application Server makes sure end-devices clocks are synchronized, and sets up, on each end-device, a fragmentation and a multicast session. When the multicast session starts, the application sends to all end-devices fragments of the firmware file leveraging the physical-layer broadcast capability of LoRaWAN. When all fragments have been received or reconstructed using error correction code, each end-device is eventually instructed to reboot using the new firmware, possibly after status and version checks.

The remainder of this paper is organized as follows. Section 2 describes the LoRaWAN FUOTA architecture. Section 3 goes through every application layer package, their purpose, and functionalities. Section 4 addresses the validation of the specifications with interoperability tests. Finally, Section 5 opens the road for future work.

2 Architecture

The LoRaWAN FUOTA architecture is depicted in Figure 1. It is composed of three elements, the LoRaWAN end-device (ED), the Network Server (NS) and the Application Server (AS).

The LoRaWAN interface, between the end-device and the Network Server is handled by the LoRaWAN link layer protocol and is by design not modified by the FUOTA specifications.

The focus of the FUOTA specifications is the File Distribution Service, composed of the multicast, fragmentation and clock synchronization basic blocks, and on the Firmware management protocol. All those packages are described in detail in Section 3.

2.1 Network Server

The Network Server is the central element handling the LoRaWAN link layer protocol. NS stands between the end-device and the Application Server. The FUOTA specifications rely on the capabilities of the LoRaWAN, and thus on the NS, to deliver the messages to the end-devices.

The NS shall follow the LoRaWAN specifications. FUOTA protocols rely specifically on the multicast capabilities of the NS. Multicast is part of the LoRaWAN protocol and requires specific care for network planning on the

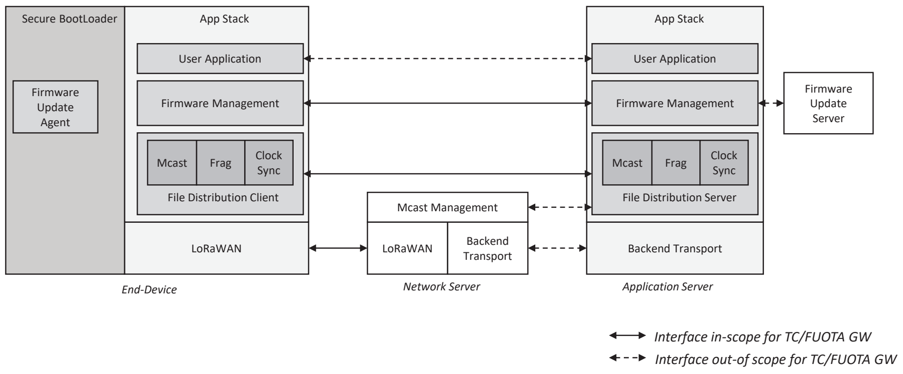

Figure 1 LoRaWAN FUOTA specifications architecture [1].

NS. The server needs to choose the set of gateways used to broadcast the messages, to reach the involved end-devices efficiently, in accordance with the local regulations and limiting self-interference.

The interface between the Application Server and the NS is not in the scope of the LoRa Alliance technical specifications and it is vendor-specific. It is often implemented using standard Internet protocol, such as HTTP REST API or MQTT clients.

2.2 End-devices

LoRaWAN end-devices supporting FUOTA need to implement, in addition to the LoRaWAN link layer protocol stack, the file distribution client composed of three protocols—namely clock synchronization, fragmentation and multicast—and the firmware management protocol. Note that, by design, those protocols can be implemented individually. They can be used for other purposes than firmware update. An example of FUOTA protocols' supplementary usage is the clock synchronization used solely for real time adjustment of the end-device clock for application needs. Another example is delivering other types of files to the end-devices, not just firmware.

The end-device architecture also includes the traditional user application. This is the essential software feature of the end-device, which lives in parallel with the FUOTA mechanism. The user application is not in the scope of the FUOTA work, except by the fact that it can be updated using the FUOTA protocols.

Another element of the end-device architecture is the firmware update agent, residing inside the bootloader, the program that initiates the device bootstrap process. This agent is not part of the specifications, as it is very specific to the end-device firmware designer and embeds the security mechanisms necessary to verify and apply the new firmware in the end-device memory.

To support the FUOTA architecture, several conditions are necessary for the end-device. Along with supporting the LoRaWAN link layer protocol, the end-device must support a multicast identity on that layer. The device must be able to store the received fragments and run the forward error correction (FEC) algorithm to recover the lost fragments. To be on-time at the multicast rendezvous, the end-device needs to keep a good track of time. To apply a new firmware, an end-device ought to verify and swap images, boot and sometimes restore another version of its firmware. All that needs to happen, in a secure fashion and using as little energy as possible, while reporting back to the Application Server.

2.3 Application Server

The Application Server (AS) is the counterpart of the end-device applicative stacks on the server side. Application Server, in its stricter sense on the LoRaWAN, handles the application encryption and decryption of the messages while also processing the application data mechanics.

As depicted in Figure 1, AS has several interfaces with the NS and the ED. As previously stated, the AS-NS interfaces are vendor-specific. The user application interface is also application provider-specific. The interface between the AS and a firmware update server is also left to the vendors.

The FUOTA work defines the interfaces between the AS and the ED for the file distribution service and for the firmware management. The set of FUOTA specifications is defining the delivery of a given file to multiple end-devices, over-the-air, with data loss protection. It also defines the basic set of commands to manage the device firmware.

There is a strong interaction between the AS and the NS. As the interfaces are vendor-specific, AS vendor must develop in accordance to NS interfaces. Moreover, as the AS needs to gather multicast settings to inform the ED, it also needs to be negotiating the parameters with the NS. On the other hand, the NS must be able to cope with the relatively large amount of data the AS will provide for broadcast. Broadcasting a firmware file is not the usual LoRaWAN downlink load NSs are used to. A typical end-device firmware file is a few tens of kilobytes in size. There is also a strong time interaction between the AS and the NS, as the ED are instructed to receive the data during a certain period of time. The AS must be sure that the NS will deliver packets within that time.

3 Application Packages

This section dives into details of each application layer packages defined to achieve FUOTA, starting with the file distribution service protocols (multicast, fragmentation and clock synchronization), then the firmware management protocol and finally the Multi-Package Access utility package.

The packages are designed to be in the application layer, with as little friction as possible with the lower layers. The definition of those packages follows the general application layer design guidelines [7]. The main guidelines are the LoRaWAN link layer version independence, the regulatory region independence, the idem-potent operations and deterministic command and answer lengths. Those guidelines serve the purpose of ease of integration and evolution of the system.

3.1 LoRaWAN Application Layer Clock Synchronization

The LoRaWAN Application Layer Clock Synchronization [2] package is designed to enable clock synchronization of the LoRaWAN end-devices with the LoRaWAN network clock, with second resolution. The LoRaWAN network clock is based on GPS clock. The GPS system is built on atomic clock, bringing high accuracy and allowing synchronization of the entire network with the same time base. In addition, the GPS clock is not affected by leap second adjustment and provides a continuous incrementing counter, starting at the so-called GPS EPOCH, on January 6th, 1980 00:00:00 UTC.

Synchronizing end-devices clock to the same time-base is a necessary feature to achieve FUOTA efficiently. The end-devices need to be well synchronized to be able to open their receiver at the same moment to catch the broadcasted frames. Better clock synchronization means less energy consumption. The end-device will not waste energy unnecessarily listening if it does not open the reception window too early. It will not waste energy reconstructing lost packets or require another broadcast session if it does not open its reception window too late.

This feature is not only useful for FUOTA but can also be used for other applications. It may be useful to have a correct notion of time to act at a given moment, for example switching on a light at sunset, or to synchronously sample sensors data with other end-devices, for example to correlate temperature reading of one sensor with a door opening sensor.

This specification is defining application layer clock synchronization, to complement the LoRaWAN link layer capabilities or other mechanisms available on the end-device. LoRaWAN link layer offers a clock synchronization procedure if the end-device is Class B capable. End-devices can also be equipped with GPS receiver and accomplish clock synchronization with that. This application package, however, does not require any additional capabilities on the end-device. It is also more power efficient, as it requires communication only if the end-device is out-of-sync. The downside of an application layer clock synchronization is the precision. The best that can be achieved with this package is around a second accuracy. This is enough for FUOTA but might not satisfy other application requirements.

The specification defines a set of commands and responses to accomplish clock synchronization. The end-device can proactively request a clock adjustment, and the network can also demand the end-device to periodically check or even force a synchronization procedure.

3.2 LoRaWAN Fragmented Data Block Transport

The LoRaWAN Fragmented Data Block Transport [3] package aims to define a means to transmit a data block, such as a firmware file of tens of kilobytes, split into fragments, to an end-device or a group of end-devices. The specification is independent of the rest of the FUOTA specifications but it can be associated with the multicast specification described in Section 3.3. The specification resides at the application layer and it does not rely on any capabilities of the underlying layers, specifically the LoRaWAN link layer frame counter or acknowledgements.

Fragmentation is a required feature for FUOTA as a firmware file is usually much larger than the LoRaWAN frame size. There is a need to define a method to split a data block in multiple fragments to fit into LoRaWAN packets. Those fragments must be reassembled on the receiver side, taking into consideration that some fragments might be lost, and that the fragment delivery may be a broadcast session to several end-devices. The requirement to perform broadcasting of fragments implies that a given end-device cannot indicate to the network when a fragment is missing, as it would create a large amount of individual uplink frames. The specification design uses Forward Error Correction (FEC) code to enable feedback-less reconstruction of lost fragments. FEC algorithms are well-known in other industries like optical data reading (Compact Disk, QR code) or video broadcast.

Fragmentation and FEC are not only useful for FUOTA operations, but also to other applications. Transferring large amount of data to an end-device can be used for configuration or cryptographic material updates.

The specification defines first the session establishment mechanisms to inform the end-devices that a fragmentation session will start, possibly associated with a multicast session. The session parameters include the expected size of the data block, the size of each fragment among other information. Cryptographic integrity of the data block is provided in the protocol. A set of commands and responses allow the network to request information on existing fragmentation sessions, to create and to delete sessions. The second part of the specification describes a forward error correction code to be able to reconstruct missing fragments. The FEC is a systematic code, i.e. the first fragments are the original ones, followed by redundancy fragments. Redundancy fragments are computed with a Low-Density Parity-Check encoding of the original fragments. When a fragment is lost and a redundancy fragment has its information, with the help of other original fragments, the data can be recovered. The protocol allows other recovery mechanisms to be defined in the future.

This specification is agnostic to the transported content. It enables splitting a data block into fragments, having the receiver(s) ready to handle the stream of data and addressing transmission loss. When the data block is received, it is up to the end-device software agent to use it in whatever purpose it was transmitted.

3.3 LoRaWAN Remote Multicast Setup

The LoRaWAN Remote Multicast Setup [4] package defines a protocol to setup and manage LoRaWAN link layer multicast sessions. LoRaWAN link layer defines two classes of communications which are multicast capable: Class B and Class C. This specification defines a means to enable a temporary Class B or Class C session. It follows the same design principle as the other specifications and resides at the application layer. There is a stronger implementation interaction with the LoRaWAN link layer and this specification. The application needs to enable some capacity at the link layer, both on the end-device and on the Network Server.

Multicast feature is essential to FUOTA as it greatly optimizes the distribution of the firmware in the network. In most cases, end-devices operator will upgrade a group of end-devices at once, not one-by-one. Multicast transmission towards multiple end-devices saves network bandwidth and time-on-air, the most precious resources of a LoRaWAN network. When a group of end-devices are within the reach of a common gateway, the radio resources used to update that group are essentially the same as upgrading a single end-device.

Multicast feature also has utility for other applications. For instance, a street lighting application will benefit from multicast to switch all of the lamps of a street using a single LoRaWAN message over-the-air.

The specification defines commands and responses to create, query and delete multicast sessions on the end-devices. A multicast session is built using two commands. One command is used to exchange the security context, specifically the LoRaWAN link layer multicast address, the valid range of frame counters and key material. A second command is specific to the communication class to enable, either Class B or Class C. That command is defining the radio parameters and the session start time and duration. It should be noted that a Class B session will last longer but would consume less power on the end-devices side, due to the slotted communication of this communication class. This is where clock synchronization is important to allow the group of end-devices to start their multicast session at the same time and in sync with the network operations.

3.4 LoRaWAN Firmware Management Protocol

The LoRaWAN Firmware Management Protocol [5] package is defining the basic operations to manage firmware on end-devices. This is the only specification of the FUOTA working group portfolio that is specific to firmware. This specification brings generic features to manage firmware, such as controlling the firmware version running on the end-device or the hardware revision of the device.

Firmware management is used from the beginning to the end of the FUOTA process. It is helpful for the network application to know the current running firmware version of each end-device before applying an update. It is useful to apply the downloaded update by rebooting the end-device.

Aside from the FUOTA process, firmware management can also be used by broader device management applications. Collecting current running firmware associated with hardware version helps fleet manager to get an accurate inventory of their devices. Being able to reboot remotely an end-device is also beneficial outside of firmware update procedure. For example, to fix a bug or have the end-device restart from a known state, reboot is sometimes necessary.

The specification defines a handful of generic commands and responses to read and act on the firmware version present on the end-device. The network application can request the currently running version of the firmware and the hardware. It can also request the ready-to-upgrade firmware version stored in the end-device memory, potentially received with a fragmented broadcast, or request to delete such image. It allows the network application to trigger a reboot or to schedule it at a later time. The notion of time is once again linked to the capacity of the end-device to be clock synchronized. Clock synchronization dependency is relaxed by this specification as it proposes relative (countdown) time operations. This allows simple end-devices to use that package without strong requirements on time keeping.

Version information provided by this package is relatively vague on purpose. It is left to the end-device manufacturer to define the content of the version fields and their meaning. Some may use the standard semantic versioning (major, minor, patch numbers), while others may encode the checksum (e.g.: CRC32) of the firmware file. The specification helps transporting that information between the end-device and the network application but does not impose any formatting.

3.5 LoRaWAN Multi Package Access Protocol

The LoRaWAN Multi Package Access Protocol [5] is defined for efficient use of multiple packages implemented on the end-devices. This protocol allows the network application to query the existing packages on a given end-device and their version. It also allows to concatenate multiple packages commands or responses in a single LoRaWAN frame. Responses are possibly fragmented by this protocol. This saves communication round trips and improve reactivity of the system. The Multi Package Access Protocol has been assigned a dedicated LoRaWAN FPort (225) so it is possible to discover the capabilities of an end-device without prior knowledge using this registered well-known port.

Multi Package Access Protocol is helpful for the FUOTA process as the procedure deals systematically with several packages. The most obvious usage of that capability is to setup the multicast and the fragmentation sessions in a single downlink frame. This avoids an additional communication round-trip which can delay the set-up. In LoRaWAN link layer Class A, a downlink only follows an uplink and the delay between uplinks is at the discretion of the end-device. The FUOTA network application can also use this access method to query the capabilities of the end-devices, especially the supported version of each package. An end-device supports only one version of a given package, and multicast, fragmentation and clock sync packages each have two versions defined already.

Querying package versions and accessing multiple packages in a single frame is useful for virtually any other application packages. In order to benefit from that feature, packages must follow simple rules [7]. They need to define opcodes, just like the the FUOTA packages do. More importantly, the command and response lengths must be deterministic and based on the opcode. If that is the case, any application package can use this protocol and be used along with FUOTA packages commands or responses.

This specification defines a command to request an end-device to provide the list of packages it runs, and their version. Each package has an identifier, a version number and a dedicated FPort. Package identifiers are registered within the LoRa Alliance for specified packages or can be end-device specific and shared out-of-band with the network application. The version number is intrinsic to the package, and the package's FPort is the direct LoRaWAN port to contact that package on the end-device. The port is defined per end-device. It can be suggested by the package definition and this protocol's discovery mechanism allows to deviate from the recommended values. A protocol is defined to access multiple packages with a single frame based on the package identifiers and their commands. Carrying multiple commands can lead to large responses, possibly larger than what LoRaWAN regional parameters specification allow. This specification defines a fragmentation method to split and request retransmission of fragments from an end-device. It is to be noted that this fragmentation is different from the one describes in Section 3.2, as it addresses uplink (from an end-device to the network application) and unicast only fragmentation, as opposed to [3] defining downlink multicast fragmentation.

4 Interoperability Tests

The FUOTA working group is chartered to define protocols presented in this document between LoRaWAN end-devices and Application Servers. In order to validate the protocol specifications, it is important that those specifications are tested and understood without ambiguity by the developers before being released to the public. There are several methods in the industry to validate a specification. Some choose to have an open-source reference implementation so that an implementer can read the specification and the source code to solve any remaining doubts.

Another way to validate a specification, especially between a device and a server, is to run interoperability tests. The idea behind those tests is to be sure that both parties, the end-device implementer and the server developer, have understood the specifications correctly and there are no details missing. This is the approach chosen by the FUOTA WG members to validate the packages. It is worth to be noted that what is evaluated is the clarity and completeness of the specification text. In case any ambiguity or misunderstanding are found, the specifications are amended before release.

Implementing interoperability tests allows members to be one step ahead as they already design and develop their products or software in advance, before the specifications are released.

5 Future Work

The previous sections have described the current state of the FUOTA capabilities designed within the LoRa Alliance. This set of specifications are relevant for firmware upgrade and to other usage as discussed. While they are enabling a standard way to achieve FUOTA capabilities on end-devices attached to a LoRaWAN network, there are many other topics to address.

FUOTA as it is defined is one block of a larger and more complete *device management* system. Being able to update software and query protocol capability is one of the challenges addressed by device management systems. They also address device configuration, device security provisioning or device monitoring. Device management systems are often associated with device data model, a method to describe capabilities of an end-device associated with an access protocol.

FUOTA set of protocols enable file distribution to a group of end-devices. Nothing is specified yet on the content of such file. Other standard bodies, such as IETF SUIT [8], are defining what should be transported to end-devices. That working group defines a manifest file format, as well as means to ensure confidentiality and authenticity of the firmware file. Another topic to address is the compression of the transferred file. Technologies to be explored include compression, delta firmware upgrade, and partial firmware upgrade algorithms. The usage of those technologies relies on the capacity of the end-device to be able to process such compression.

Ultimately, not everything is relevant to be standardized, as the end-device application/business logic and firmware format will be deeply dependent on the manufacturer and application requirements.

References

[1] FUOTA Working Group of the LoRa Alliance Technical Committee, FUOTA Process Summary Technical Recommendation TR002, Jan. 2019.

[2] FUOTA Working Group of the LoRa Alliance Technical Committee, LoRaWAN Application Layer Clock Synchronization Specification v1.0.0 TS003-1.0.0, Sep. 2018.

[3] FUOTA Working Group of the LoRa Alliance Technical Committee, LoRaWAN Fragmented Data Block Transport Specification v1.0.0 TS004-1.0.0, Sep. 2018.

[4] FUOTA Working Group of the LoRa Alliance Technical Committee, LoRaWAN Remote Multicast Setup Specification v1.0.0 TS005-1.0.0, Sep. 2018.

[5] FUOTA Working Group of the LoRa Alliance Technical Committee, LoRaWAN Firmware Management Protocol Specification v1.0.0 TS006-1.0.0, to be released.

[6] FUOTA Working Group of the LoRa Alliance Technical Committee, LoRaWAN Multi Package Access Specification v1.0.0 TS007-1.0.0, to be released.

[7] FUOTA Working Group of the LoRa Alliance Technical Committee, Application Layer Package Design Guidelines Technical Recommendation TR003, to be released.

[8] Internet Engineering Task Force (IETF), Software Update of Internet of Things (SUIT) working group, work in progress.

Biography

Julien Catalano is Principal Architect and Head of Standards working at Kerlink in France since 2013. Julien also acts as chair of the Technical Committee FUOTA Working Group at the LoRa Alliance since 2017.

He holds a master's degree from the École Centrale de Marseille and started his career as a research engineer for Philips Research in Eindhoven, the Netherlands.

Julien has more than 15 years of experience in the Internet of Things industry, tackling embedded software development, system architecture and strategic partnership in research, industrial companies and start-ups. He is a strong believer in open standards, building them in several organizations. This sparked his recognition by the LoRa Alliance with the Working Group Leadership Award in 2019.