Enabling firmware updates over LPWANs

Jan Jongboom
Internet Services Group, Arm
Amsterdam, the Netherlands
jan.jongboom@arm.com

Johan Stokking
The Things Industries
Amsterdam, the Netherlands
johan@thethingsindustries.com

***Abstract*—Firmware updates are essential for large-scale deployment of connected devices. Security patches protect customer and business data, and new functionality, optimization and specialization extend the lifetime of devices. This paper discusses firmware updates over the most challenging type of networks: low-power and long-range networks.**

Keywords—LPWAN, LoRaWAN, Sigfox, NB-IoT, firmware updates

I. Introduction

Most Internet of Things (IoT) devices require both long-range and low-power consumption where a battery life can last years. Traditional wireless network technologies, such as cellular and Wi-Fi, cannot accommodate these needs. To facilitate the requirements of these devices, new network technologies called "Low-Power Wide Area Networks" (LPWANs) have emerged in the past few years. Networks such as LoRaWAN, Sigfox and NB-IoT are deployed using low-cost radio chips with kilometers of range and low battery consumption.

A downside of these networks is that the data rates are much lower than those of traditional radio networks. Data rates in LPWANs are measured in bits per second, rather than megabytes per second. Additionally, many of these networks operate in the unlicensed spectrum (ISM band), which requires devices to adhere to duty cycle limitations, only allowing them to send a fraction of the time while suffering from interference. These characteristics make it difficult to support firmware updates over the air. This implies you cannot update the devices deployed in the field easily: especially devices that are deployed in places that are almost impossible to reach, or where the cost of sending a technician is too high with thousands of devices in a variety of places.

Not being able to update the firmware on IoT devices easily is an extreme challenge when deploying at scale. First, you can never have 100% secure software; we have seen many examples of this in 2017. Second, these devices may have to last up to ten years, so keeping them up to date with the latest standards and protocols is important. Lastly, the ability to add functionality or specialize devices throughout the lifetime, from manufacturing and distribution to transfer of ownership or change of purpose, is critical in many business cases.

The key requirements for firmware updates are the abilities to:

1. Send data to multiple devices at the same time (so called multicast) in an efficient manner in terms of power consumption and channel utilization.
2. Recover from lost packets.
3. Verify the authenticity and integrity of the firmware while following standards end-to-end.

This article discusses these challenges one by one and presents a solution.

II. Multicast

Unlike cellular or Wi-Fi, with which a device maintains a connection with the network at all times, most LPWANs are uplink oriented. Sending data (uplink) is more important than receiving data (downlink). It is only possible to send a downlink message at set times, which LPWAN refer to as RX windows. These RX windows only open shortly after a transmission, which is great for battery life because the device does not need to maintain the connection with the network and can go to sleep mode as much as possible. LoRaWAN Class-A, Sigfox and LTE-M in Power Save Mode (PSM) follow this model.

However, for sending firmware images, this is terrible because you need downlink oriented transmission of many packets. With a payload size of 115 bytes taking 615 ms. air time (a typical transmission speed for LoRaWAN[1]), you need to exchange 891 messages to send a 100 KB firmware image. Because of the 1% duty cycle requirement in many markets (including Europe), this requires over 9 hours to update a single device, assuming no packet loss. In addition, the gateways may cover hundreds or thousands of devices that are also subject to duty cycle limitations, which means it may take weeks to update a fleet of devices. Finally, for every received packet, the required transmission consumes lots of energy (typical LPWAN TX consumes ~50 mA, and RX ~9 mA current) and uses a lot of the available spectrum.

To enable more efficient firmware update capabilities, you need to implement two features:

1. A way to send the firmware image without the device requiring to transmit first, optimizing the device's duty cycle and power consumption.
2. Multicast support - for updating multiple devices at the same time, optimizing the gateway duty cycle.

The first step is to get all devices that you need to update to listen at the same time, at the same frequency, data rate and security session. If you load the same keys to the devices, then all devices can both receive and decrypt the same packets as if they are one device. Once you are certain that the devices are listening, you can start broadcasting the firmware image without the need for devices to transmit first. This means that you need to schedule firmware updates, typically hours or days in advance, depending on the sleep behavior of the devices requiring the update.

Because the network can continuously send messages, you can transmit the 891 packets (100 KByte) in under six minutes (at 600 ms. time on air per packet). Multicast firmware updates are only achievable for stationary devices because the gateways selected and the data rate need to be fixed when you create the multicast session, which can be days before the session starts.

III. Network Reliability

By doing this, there is an effect on the network reliability. The network still needs to adhere to the duty cycle limitation of the gateways, which send the packets. This means that a firmware update can render a gateway useless for relaying downlink messages for a while. A way of mitigating this is to have coverage by multiple gateways, and round-robin[2] between gateways during the update. Also, because most LPWAN gateways are only half-duplex, devices cannot use the frequency that the update uses. This is not such a problem on licensed spectrum (NB-IoT / LTE-M1), or in areas with wide unlicensed spectrum available (U.S.), but it is in Europe where LPWANs are deployed in limited spectrum (LoRaWAN only has eight channels in the EU). A way of mitigating this is to implement frequency hopping. (Weightless-N does this.)

Another way is for a technician to drive to the deployment site with a separate gateway and use it for the update. Although this seems unnatural, it has some benefits when deploying on a constrained site. Because the reception is more predictable, you can use a higher data rate when sending the update, which causes less congestion on the network. Additionally, the update does not affect normal gateways. This could be an option for hard-to-reach sensors. This method is also an option for LPWANs that have a smaller link budget for downlink messages than for uplink messages, such as Sigfox.

IV. Security for Firmware Update over Multicast

When instructing multiple devices to join a temporary multicast session in which all the devices share the same session keys, there is a potential security risk when one of the devices is compromised: packet injection. This is applicable because most LPWANs choose a symmetric authentication mechanism. Having the multicast session keys, the attacker can send packets as if they came from the server. Although this is a serious issue when using multicast without additional security measurements, such as controlling lights simultaneously, a firmware update mechanism requires three additional measures to secure the update process.

First, once the device receives the file, it calculates the checksum of the data it received. The device sends this checksum to the network using its private secure session. The server compares this checksum with the checksum of the data that it sent. This check fails if the data has been tampered with. The server responds whether the checksum is correct to each device individually, on its private secure sessions.

Second, as part of the server's response to indicate the correctness of the checksum, the server sends the message integrity code (MIC), which guarantees data integrity to the device. No one who does not know the device's private secure session keys can forge this MIC: only the device and the server can calculate the same MIC. So the server checks the device's checksum, and the device checks the server's MIC, communicating on the device's private secure session.

Third, when an attacker injects random packets, the device may not be able to reconstruct the original image. To avoid devices that run out of power because they keep listening for error correction packets, as presented in the next section, the multicast session should have a lifetime: a fixed limit on the number of messages. When reaching this limit, the device switches back to its private secure session and power efficient operating mode, and discards all data.

V. Sending Large Binary Packets over a Lossy Network

In the schema proposed above, there is no communication between the device and the network when the multicast transmission is in progress. Thus, it is not possible to determine which device received which fragments of firmware update. This is required to minimize spectrum usage, similar to how UDP works. On LPWAN networks, there is often no guaranteed quality of service, and packet loss can occur when the device is moving. To deal with the high packet loss, you should implement an error-correcting algorithm, which does not require communication from the device to the network. One such algorithm is Low-Density Parity Check Coding[3].

In the first step, the network sends the firmware as is, fragmented in packets. Next, the network starts sending error correction packets, which are XORed to what the device already received. Because the fragments have an increasing frame number, the device knows which fragments are missing and can use the correction packets to reconstruct the missed fragments. The network keeps sending correction packets until all devices confirm that they reconstructed all the fragments of the firmware update, or, in case of extreme packet loss, until the update server sent all correction packets. With a good error correction algorithm, you need up to five correction packets to correct for three missed fragments.

After the device reconstructs the full firmware, the device switches back to its private secure session and operating mode. After successfully testing the device's checksum and the server's message integrity code as presented above, the device performs the firmware update.

DELTA UPDATES

To minimize the amount of data sent, it's advisable to implement delta updates, in which the network sends only the changed parts of the firmware image instead of the full image. This can reduce the data by 90%. For constrained devices, it's important to choose a linear patch format, which you can apply while using little memory. In addition, it's important to verify the authenticity of the firmware after patching to avoid bricking devices.

VI. CRYPTOGRAPHIC VERIFICATION OF THE FIRMWARE

The devised protocol only handles raw data integrity of the firmware. It involves timing and message level security and accounts for packet loss. However, a good firmware update process also requires additional security on top of the network layer because hijacking the firmware update algorithm is a big attack vector.

To protect against these attacks, you need to program extra properties into the end device:

- A public key of the owner who is authorized to update firmware on the device.
- A manufacturer universally unique identifier (UUID).
- A device type UUID.

The firmware update should contain a manifest that consists of the cryptographic hash of the update, the manufacturer and the device type that the update applies to, all signed with the manufacturer's private key. Whenever the device receives the update, you can verify that a trusted authority signed it and whether it was meant for this device because the device contains the manufacturer's public key.

For the cryptographic hash, we suggest you use at least single-curve ECDSA/SHA256[4], which you can efficiently implemented on constrained devices while still providing adequate security.

VII. CONCLUSION

Firmware updates are an essential requirement before devices that use LPWANs for connectivity hit the market in volume. When implementing the requirements in this paper, device manufacturers can ship products while assuring their customers of security updates, new functionality, optimizations and specialization throughout the device's lifetime.

To demonstrate that multicast firmware updates are possible, Arm and The Things Industries have developed a reference implementation on top of LoRaWAN that implements the suggestions from this paper. The result is a secure, fast and efficient method of updating constrained devices (under 32K RAM required) in the field. You can implement the reference implementation on other LPWANs, too, and it is licensed under the permissive Apache 2.0 (device firmware) and MIT (network server) license. You can find more information at https://mbed.com/fota-lora.

REFERENCES

[1] https://www.thethingsnetwork.org/forum/t/spreadsheet-for-lora-airtime-calculation/1190

[2] https://en.wikipedia.org/wiki/Round-robin_scheduling

[3] https://en.wikipedia.org/wiki/Low-density_parity-check_code

[4] https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm