A Firmware Update Architecture for Internet of Things Devices

draft-ietf-suit-architecture-01

Changes between -00* and -01

- New terminology for entities
- Updated operating modes
- Device Firmware Update Examples
- Added David Brown as co-author
- Many editorial changes
- New figures

*: draft-ietf-suit-architecture-00 was discussed at the virtual interim meeting

ENTITIES

Author & Device

- **Author**: The author is the entity that creates the firmware image and a manifest.
  - There can be multiple authors in a system (□ firmware consisting of multiple software components, or device running multiple MCUs)
  - There are also other parties that can create a manifest even though they do not create new firmware
- **Device**: Definition updated to point out that the device may need multiple firmware images.

Communicator

- The communicator component of the device interacts with the firmware update server.
- It receives firmware images and triggers an update, if needed.
- The communicator either polls a firmware update server for the most recent manifest/firmware or manifests/firmware images are pushed to it.
- Note that the firmware update process may involve multiple stages since one or multiple manifests may need to be downloaded before the communicator can fetch one or multiple firmware images/software components.

Status Tracker

- The status tracker offers device management functionality that includes keeping track of the firmware update process. (It typically knows what firmware / software is run on the devices.)
- This includes fine-grained monitoring of changes at the device, for example, what state of the firmware update cycle the device is currently in.

Firmware Server

- Entity that stores firmware images and manifests. Some deployments may require storage of the firmware images/manifests on more than one entities before they reach the device.

Device & Network Operator

- Device Operator: The actor responsible for the day-to-day operation of a fleet of IoT devices.
- Network Operator: The actor responsible for the operation of a network to which IoT devices connect.
- Both may also create manifests (for already existing firmware)

Trust Provisioning Authority (TPA)

- The TPA distributes trust anchors and authorization permissions to various entities in the system.
- The TPA may also delegate rights to install, update, enhance, or delete trust anchors and authorization permissions to other parties in the system.
- *This infrastructure overlaps the communication architecture and different deployments may empower certain entities while other deployments may not.*

OPERATING MODES

Operating modes

1. **Client-initiated Update**:
   Client-initiated updates take the form of a communicator on a device proactively checking for new firmware imagines provided by firmware servers.
2. **Server-initiated Update**:
   The status tracker determines what devices qualify for a firmware update. Once those devices have been selected the firmware server, in cooperation with the status tracker, distributes updates to those devices.
3. **Hybrid Update**:
   The status tracker pushes notifications of availability of an update to the device, and the communicator then downloads the image from the firmware server when it wants.

COMMUNICATION ARCHITECTURE

Communication Architecture

```
             Firmware +  +----------+       Firmware + +-----------+
             Manifest    |          |-+     Manifest   |           |-+
              +--------->| Firmware | |<---------------|           | |
              |          | Server   | |                |  Author   | |
              |          |          | |                |           | |
              |          +----------+ |                +-----------+ |
              |            +----------+                  +-----------+
              |
              |
              |
             -+--                                  ------
        ----  |  ----                          ----      ----
      //      |      \\                      //              \\
     /        |        \                    /                  \
    /         |         \                  /                    \
   /          |          \                /                      \
  /           |           \              /                        \
 |            v            |            |                          |
 |     +------------+                                              |
 |     |Communicator|      |            |                          |
|      +--------+---+       | Device    |       +--------+          |
|      |        |           | Management|       |        |          |
|      | Device |<----------------------------->| Status |          |
|      |        |           |          |        | Tracker|          |
|      +--------+           |          ||       |        |         |
 |                         |           ||       +--------+         |
 |                         |            |                          |
 |                         |             \                        /
  \                       /               \                      /
   \                     /                 \      Device        /
    \     Network       /                   \     Operator     /
     \   Operator      /                     \\              //
      \\             //                        ----      ----
        ----     ----                              ------
            -----
```

DEVICE FIRMWARE UPDATE EXAMPLES

Single CPU SoC

- The simplest, and currently most common, architecture consists of a single MCU along with its own peripherals.
- These SoCs generally contain some amount of flash memory for code and fixed data, as well as RAM for working storage.
- These systems either have a single firmware image, or an immutable bootloader that runs a single image.
- A notable characteristic of these SoCs is that the primary code is generally execute in place (XIP).
- Combined with the non-relocatable nature of the code, firmware updates need to be done in place.

Single CPU with Secure - Normal Mode Partitioning

- Another configuration consists of a similar architecture to the previous, with a single CPU.
- However, this CPU supports a security partitioning scheme that allows memory (in addition to other things) to be divided into secure and normal mode.
- There will generally be two images, one for secure mode, and one for normal mode. In this configuration, firmware upgrades will generally be done by the CPU in secure mode, which is able to write to both areas of the flash device.
- In addition, there are requirements to be able to update either image independently, as well as to update them together atomically, as specified in the associated manifests.

Dual CPU, shared memory

- This configuration has two or more CPUs in a single SoC that share memory (flash and RAM). Generally, they will be a protection mechanism to prevent one CPU from accessing the other's memory. Upgrades in this case will typically be done by one of the CPUs, and is similar to the single CPU with secure mode.

Dual CPU, other bus

- This configuration has two or more CPUs, each having their own memory.
- There will be a communication channel between them, but it will be used as a peripheral, not via shared memory. In this case, each CPU will have to be responsible for its own firmware upgrade.
- It is likely that one of the CPUs will be considered a master, and will direct the other CPU to do the upgrade.
- This configuration is commonly used to offload specific work to other CPUs.
- Firmware dependencies are similar to the other solutions above, sometimes allowing only one image to be upgraded, other times requiring several to be upgraded atomically. Because the updates are happening on multiple CPUs, upgrading the two images atomically is challenging.

NEXT STEPS

Random Thoughts

- More editorial clean-up
- Incorporate feedback from this meeting
- More text about device interactions and bootloader design
- Better alignment with information model