Draft Recommendation ITU-T X.secup-iot

Secure software update for Internet of things devices

Summary

Recommendation X.secup-iot specifies: 1) basic models and procedures for securely updating Internet of things (IoT) software/firmware (SW/FW) and 2) requirements and capabilities for updating IoT FW.

A common secure update procedure is specified with general requirements. This procedure allows common IoT SW/FW updates to be securely implemented among stakeholders in the IoT environment, such as IoT device developers and IoT system/service providers.

Keywords

Software update; IoT; security.

Table of Contents

1 Scope 4

2 References 4

3 Definitions 4

3.1 Terms defined elsewhere 4

3.2 Terms defined in this Recommendation 4

4 Abbreviations and acronyms 4

5 Conventions 5

6 Basic Model 5

7 Update procedures 5

8 Deployment Scenarios 6

8.1 Functional entities inside IoT devices 6

8.2 Cascading status trackers 6

8.2.1 A status tracker inside an IoT device directly communicates with an FW-server 7

8.2.2 A status tracker inside an IoT device and one on the Intranet work together 7

8.2.3 Multiple status tracker s work together 8

9 Discovery of available new firmware images and initiation of the procedure 8

10 Requirements 9

11 Capabilities 9

11.1 Capabilities of an firmware consumer 10

11.2 Capabilities of a status tracker 10

11.3 Capabilities of a firmware server 10

11.4 Capabilities of an author 11

12 Security Profiles 11

Appendix 1: Related activities outside ITU-T 12

Appendix II: An example scenario of IoT software update using distributed ledger technology 13

II.1 Overview 13

II.2 Software update procedure 13

Bibliography 15

Introduction

Cyber attacks against Internet of things (IoT) devices/systems are becoming increasingly sophisticated, intelligent and varied. Previously, the functions of most IoT devices were fixed by IoT device vendors in their initial release phase. However, recently, devices are connected to the Internet to provide an enhanced set of IoT services. Therefore, IoT devices in use are facing cyber threats/attacks and it needs to be recognized that software/firmware (SW/FW) implemented in the IoT need to be securely updated to fix their vulnerabilities and weaknesses. Some device vendors have already started providing an FW update service, but such vendors provide update services by means of their own schemes.

This Recommendation provides basic models and procedures for securely updating IoT SW/FW, as well as associated requirements and capabilities. With the basic models and the common update procedure, IoT SW/FW can be securely exchanged among stakeholders in the IoT environment with encouragement to update outdated IoT SW/FW.

Draft Recommendation ITU-T X.secup-iot

Secure software update for Internet of things devices

1 Scope

This Recommendation specifies basic models and procedures for securely updating IoT software/firmware (SW/FW). It also describes requirements and capabilities for IoT SW/FW updates.

2 References

None.

3 Definitions

3.1 Terms defined elsewhere

None.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

1.1.1 **Author:** An entity that produces FW images and SW for IoT devices. Examples include an individual or a group, such as a company or any other type of organization. It may upload the image to one or more FW servers that are not necessarily trusted.

1.1.2 **Firmware consumer:** An entity that stores, verifies, and runs FW images on an IoT device. It should decide whether to run the current FW images. An IoT device has one or more FW consumers.

1.1.3 **Firmware server:** An entity that distributes FW images. It could accept FW images from multiple author; it could be a repository for a particular vendor or a repository that accepts various vendors. Ideally, an FW server is trustful, but it could be untrusted; it could try to view or modify the FW packages received from authors.

1.1.4 **Manifest:** A record that contains metadata of an FW image

1.1.5 **Status tracker:** An entity that checks and keeps tabs on the status of the FW images inside one or more FW consumers and initiates the FW updates as needed. This includes fine-grained monitoring of changes at the device, e.g., the version of the running FW images and the state of the FW update cycle the device is currently in. A status tracker may reside inside an IoT device, on the Intranet, or on the Internet.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

DLT: Distributed Ledger Technology

FW: Firmware

SW: Software

5 Conventions

None.

6 Basic Model

The network architecture of IoT devices can differ, but four functional entities are required in all the cases, i.e., FW-consumer, status tracker, author and FW-server. See clause 3.2 for their definitions. Note that multiple functional entities can reside inside one node. For instance, a webcam device contains multiple FW-consumers and a status tracker, while a web server contains a status tracker and an FW-server. Multiple FW-consumers can reside inside one network and are monitored by a status tracker implemented inside the gateway. Depending on the degree of constraints on the IoT devices, such design may differ. Typical deployment models are described in clause 8.

In the basic model, these entities play indispensable roles in updating IoT SW/FW. The basic scenario in this model is simple; *a status tracker that recognizes the need for an IoT SW/FW update initiates the SW/FW update procedure that allows FW-consumers to receive a SW/FW image from an author through an FW-server*.

7 Update procedures

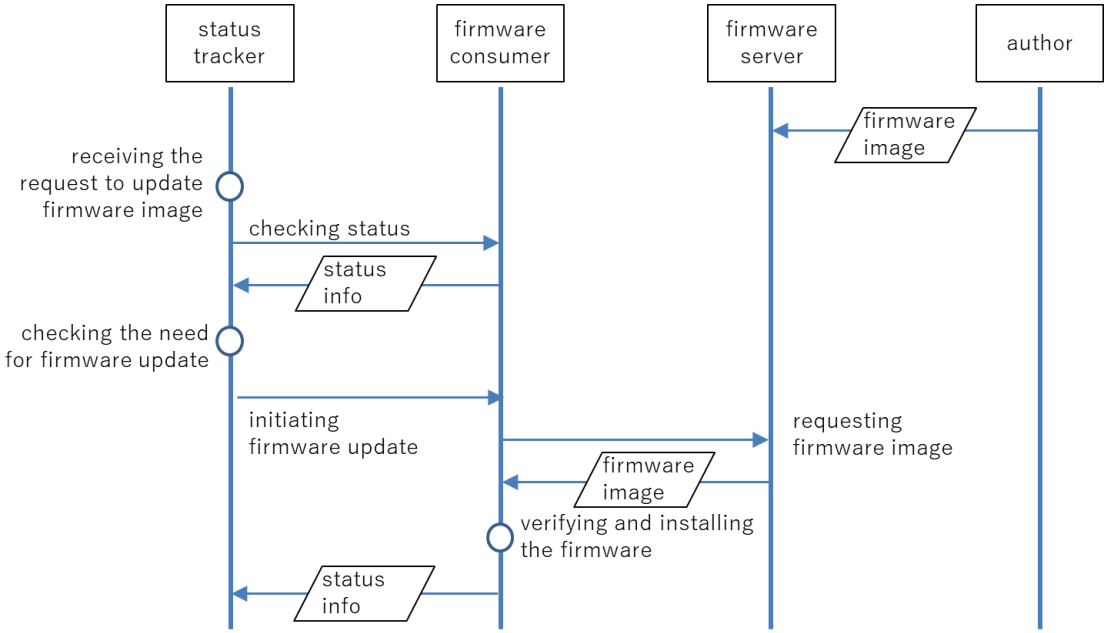

Figure 1 Protocol Procedure

Figure 1 depicts the basic procedure for updating the FW. Prior to initiating the FW update procedure, an author needs to upload a new FW image to an FW server. It is desirable that the image is accompanied with a digital signature and is encrypted by the author.

When a status tracker receives the request to update an FW image with its location [e.g., its uniform resource locator (URL)], it verifies the request and then, if the request is valid, checks the status of the FW by communicating with the FW-consumer in order to confirm the need for an FW update. Some typical ways of sending such requests are listed in clause 9.

If the need to update the FW is verified, the FW-consumer initiates an FW update by informing the location of the available FW. The FW-consumer then requests the updated FW image from the FW

server. The FW-server provides the FW image to the FW-consumer provided that the FW-consumer has the legitimate right to receive the update. Otherwise, the server sends an update message with an error code.

On receiving the update message, the FW-consumer verifies the image. If no error is found, the FW-consumer installs the FW and sends status information to the status tracker. Note that the cardinalities of the above four functional entities are many to many, i.e., multiple status trackers may communicate with multiple FW-consumers, which may communicate with multiple FW-servers, which may communicate with multiple authors.

8 Deployment Scenarios

As mentioned in clause 6, multiple functional entities can reside inside one node, and multiple entities can serve as a functional entity; the deployment scenarios may differ depending on cases. In this clause, several deployment scenarios are illustrated.

8.1 Functional entities inside IoT devices

Figure 2 shows four types of IoT devices. An IoT device must contain at least one FW-consumer because it is natural that an IoT device contains multiple FW images.

An IoT device must contain at least one status tracker. It could contain multiple status trackers to handle multiple FW-consumers, but having a single status tracker that handles all of the FW-consumers also works fine.

A resource-constrained IoT device may wish to minimize the functionality of the status tracker. In this case, some functionality of the status tracker is outsourced to another status tracker (i.e., one upstream) deployed outside the IoT device. However, the IoT device still needs a status tracker to receive the trigger to initiate the FW update procedure, to verify the trigger and to check the status of the FW-consumer(s) inside the IoT device.

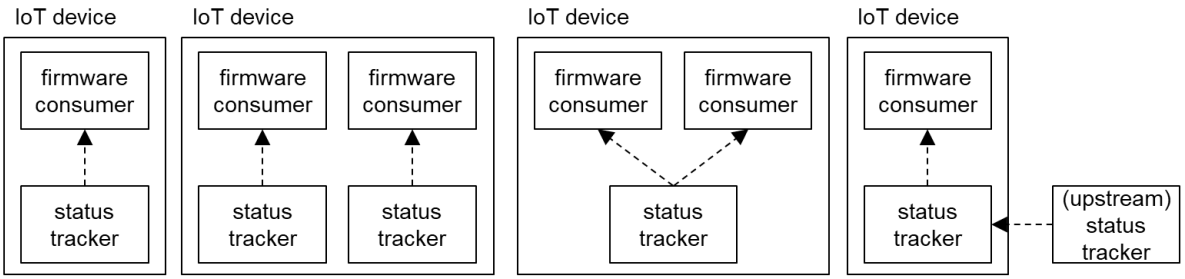

Figure 2 Different types of IoT devices

8.2 Cascading status trackers

The resources of IoT devices differ a lot; a status tracker may reside inside a status tracker, but resource constrained IoT devices may wish to keep it separately and minimize its resource consumption. There could be a case where the IoT devices are preferred to be managed by a centralized status tracker entity for the ease of management.

To cope with these situations, a status tracker may be implemented in a hierarchical manner. In this case, multiple status trackers can be cascaded so that upstream status tracker may judge the need for FW update and initiate the FW update procedure through the downstream status trackers.

Cases where (1) a status tracker inside an IoT device directly communicates with an FW-server, (2) a status tracker inside an IoT device communicates with an FW-server via another status tracker residing inside the Intranet, and (3) a status tracker inside an IoT device communicates with an FW-server via multiple status trackers are illustrated in clause 8.2.1 to 8.2.3.

8.2.1 A status tracker inside an IoT device directly communicates with an FW-server

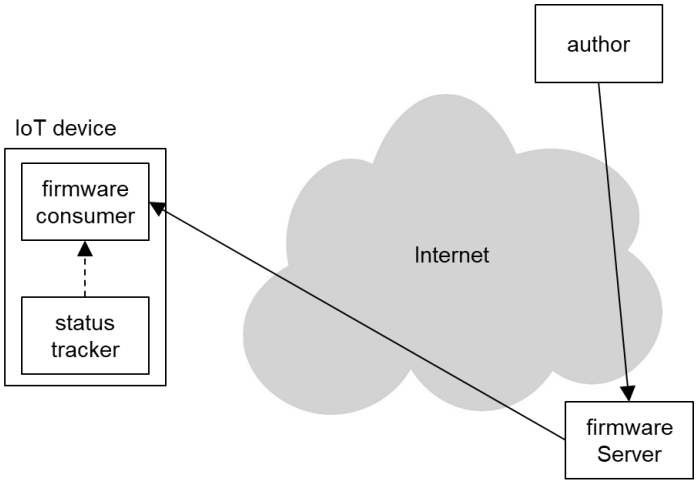

Figure 3 IoT device directly connected to the Internet

Figure 3 depicts a scenario in which an FW-consumer and a status tracker reside inside an IoT device that is directly connected to the Internet. When the status tracker realizes the need for the FW update, it asks the FW-consumer to receive FW images from the FW-server.

8.2.2 A status tracker inside an IoT device and one on the Intranet work together

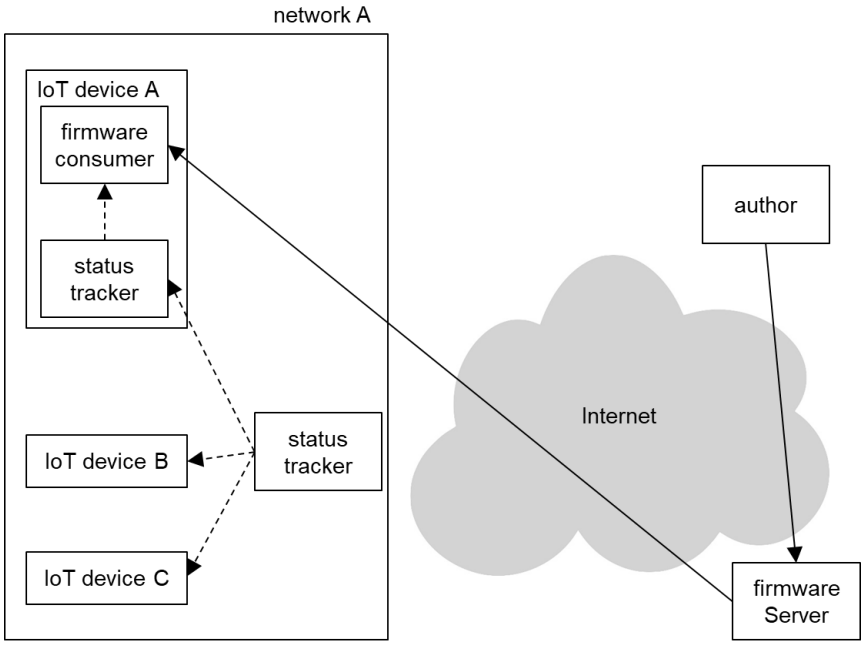

Figure 4 A status tracker takes care of multiple IoT devices

Figure 4 depicts a scenario in which a status tracker inside a network monitors several IoT devices. The status tracker inside IoT devices simply verifies the message from an upstream status tracker and acts accordingly. It can be seen that a functional entity of the status tracker is realized through the cooperation between the upstream status tracker and downstream status tracker. The upstream status tracker initiates the FW update procedure.

8.2.3 Multiple status tracker s work together

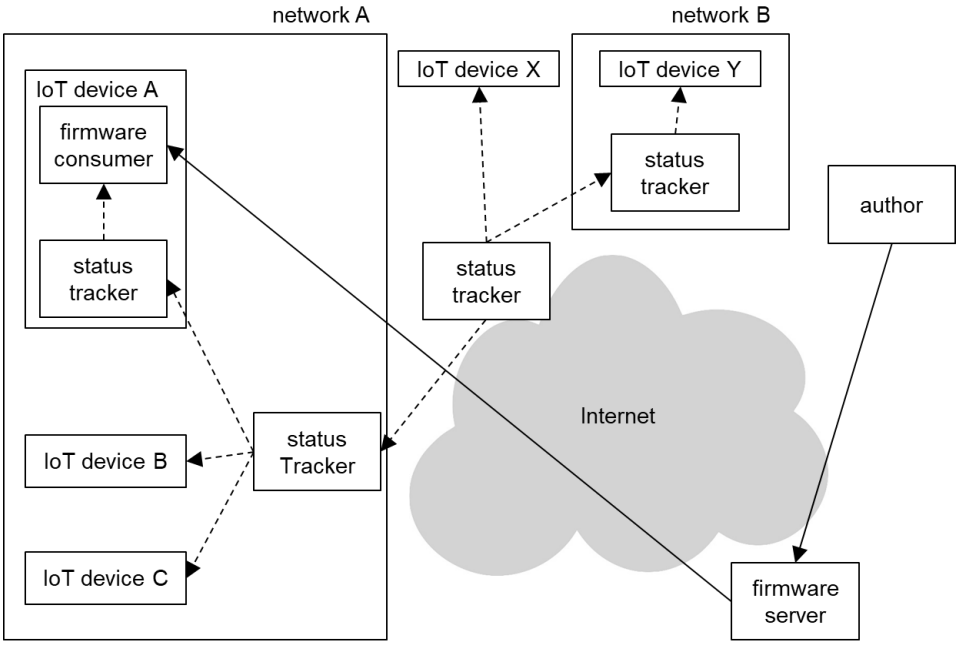

Figure 5 A status tracker takes care of multiple IoT devices inside multiple networks

Figure 5 depicts a scenario in which a status tracker monitors IoT devices inside multiple networks. Multiple status trackers exist, and a functional entity of a status tracker is realized through co-operation between upstream status trackers and downstream status trackers. The most upstream status tracker initiates the FW update procedure.

9 Discovery of available new firmware images and initiation of the procedure

The whole process is initiated by a status tracker when it receives a request to update the FW image. This type of request can take various forms:

a) An author who publishes new versions of FW images may send the request
b) An FW-server who receives new version of an FW image may send the request
c) An IoT device administrator recognizes the release of a new version of the FW images and may send the request.
d) A status tracker or one of its upstream status trackers discovers the new version of the FW images by periodically polling the FW-server
e) A status tracker or one of its upstream status trackers recognizes the existence of new version of the FW image by observing an FW update procedure of another IoT device that it takes care of

Various other events may issue the requests, but those requests need to deliver information on the URL of the FW images and their versions to the status tracker. If the status tracker judges that the information is reliable and trustful, it can initiate the procedure mentioned in clause 7.

10 Requirements

In this clause, the functional requirements for IoT SW updates are listed. Due to resource constraints, not all SW update procedures available in a non-constrained environment are applicable. It is often the case that no human user or operator is near the IoT devices. Therefore, when designing the concrete security update procedure, these differences need to be taken into account. Note that confidentiality, integrity and availability of the four functional entities must be preserved, and these are prerequisites for the SW update; thus they are omitted in clauses 10.1 to 10.5.

a) Malicious SW/FW must not be distributed
   i) Malicious images should be identified before being uploaded or exchanged
   ii) Integrity of FW images must be verifiable
   iii) The provider of FW images must be verifiable

b) Vulnerable SW/FW must not be left without any proper measures
   i) Obsolete version of SW/FW should be detected
   ii) Vulnerable SW/FW should be detected

c) Failure caused during the update procedure must be recoverable
   i) If a SW/FW update fails, there should be a means to know the situation
   ii) There should be a fallback means or protection means in the case of update process failure

d) Only intended and necessary updating should be conducted
   i) Only the newer versions of IoT SW/FW are allowed to installed
   ii) Only trusted IoT SW/FW images are allowed to be installed

e) Resource constraints need to be considered
   i) An update procedure should not occur if there is no need for it to minimize network resources
   ii) status tracker functions can be cascaded to minimize the burden of resource-constrained IoT devices

f) The intellectual property rights of authors must be preserved
   i) The SW/FW images must be encrypted by authors
   ii) The confidentiality, integrity, and availability of IoT SW/FW images must be preserved

11 Capabilities

Based on the requirements mentioned in clause 10, the capabilities of functional entities are listed in this clause.

11.1 Capabilities of an firmware consumer

a) An FW-consumer should be capable of
   i) verifying whether the previous run of an FW update was successful
   ii) sharing the information on its current SW/FW images (e.g., version number) with the parties who request this information with legitimate rights
   iii) implementing a fallback means or protection means in the case of update process failure
   iv) notifying the status tracker of the need for an FW update
   v) verifying the authenticity and integrity of FW images (e.g., by verifying their certificates)
   vi) choosing not to installing the new version of SW/FW images
b) An FW-consumer is recommended to have a “safe mode” that runs the IoT device with minimal functionality and that at least provides a means to manually install/restore/update FW

11.2 Capabilities of a status tracker

A status tracker should be capable of

a) maintaining lists of FW-consumers, those with updated FW images and those with obsolete ones.
   i) Those lists should minimally contain their unique identifiers.
   ii) A status tracker should be able to identify FW-consumers with obsolete FW.
b) knowing the status of the FW-consumers under its administration
   i) There must be a means to confirm whether the FW-consumer (and the IoT device that contains the FW-consumer) is up by communicating with the FW-consumer and by checking the communication logs with the FW-consumer.
   ii) There should be a means to confirm whether a previous run of an FW update at an FW-consumer was successful.
   iii) There should be a means to know the versions of FW that FW-consumers run.
c) judging whether an SW/FW update is necessary and initiating FW update procedure upon needed.
d) verifying the authenticity of an upstream status tracker when implemented in a hierarchical manner

11.3 Capabilities of a firmware server

An FW server should be capable of

a) accepting submissions of IoT SW/FW images from authors.
b) providing SW/FW images it contains to FW-consumers.

c) identifying malicious SW/FW images and taking appropriate action, such as removing them from its internal storage and banning submission from authors who submit such images.

d) managing the list of authors and FW-consumers that use it.

e) managing versions of IoT SW/FW images.

f) maintaining the list of FW-consumers and the SW/FW images they downloaded in the past.

g) notifying the IoT devices that have downloaded obsolete IoT SW/FW images in the past of the availability of new versions.

h) checking the geographical or logical location of IoT devices and allow or deny downloading SW/FW images in order to avoid distributing them in forbidden location identified by policies or the other means.

11.4 Capabilities of an author

An author needs to maintain the authenticity, confidentiality, and integrity of the FW images it produced.

a) FW must not be replaced or compromised by third parties (authenticity and integrity).

b) The intellectual property of vendors within the FW must be preserved (confidentiality).

c) An author should implement measure to secure the FW image it uploads to FW-servers because an FW-server is not necessarily trusted.

12 Security Profiles

Several security profiles may be prepared. The security levels required differ depending on the capability and environment of IoT devices. With the profile, appropriate FW packages can be selected and installed/updated. For instance, a boot loader may calculate the FW size and memory size and choose appropriate FW packages to install/update.

Appendix 1: Related activities outside ITU-T

(This appendix does not form an integral part of this Recommendation.)

Activities related to IoT SW updates that have been studied outside ITU-T include:

1) IOTSU workshop [b-ISOC iotsu]
2) IETF SUIT working group [b-IETF suit]
   - On manifest file [b-IETF manifest]
   - On FW update architecture [b-IETF architecture]
3) oneM2M: Standards for M2M and the Internet of Things [b-oneM2M]

…

etc.

Appendix II: An example scenario of IoT software update using distributed ledger technology

(This appendix does not form an integral part of this Recommendation.)

II.1 Overview

IoT infrastructure is feature in that the number of device that an administrator needs to manage is very large. And IoT device has various revisions according to the hardware feature, and different FW may be applied according to the additional sensor board. And depending on the hardware version, the supported SW version differs. And the differences in installed SW packages can cause dependency problems. This problem can be solved by using distributed ledger technology.

Distributed ledger technology has a smart contract that allows hardware FW or SW updates based on contracts pre-written by the administrator. And security vulnerabilities that can occur during the update process can be solved based on consensus algorithms and cryptography layer

This example describes how to provide secure FW/SW updates based on distributed ledger in environments with different hardware revisions and SW versions, such as IoT infrastructure.

II.2 Software update procedure

Table II-1. Block structure for SW update.

Block header	- Block size, version - Previous block header hash
Block data	- Merkle root - Name of provider, publish time, version number, hash code of the file, file link, filename, file size, support hardware, SW dependency

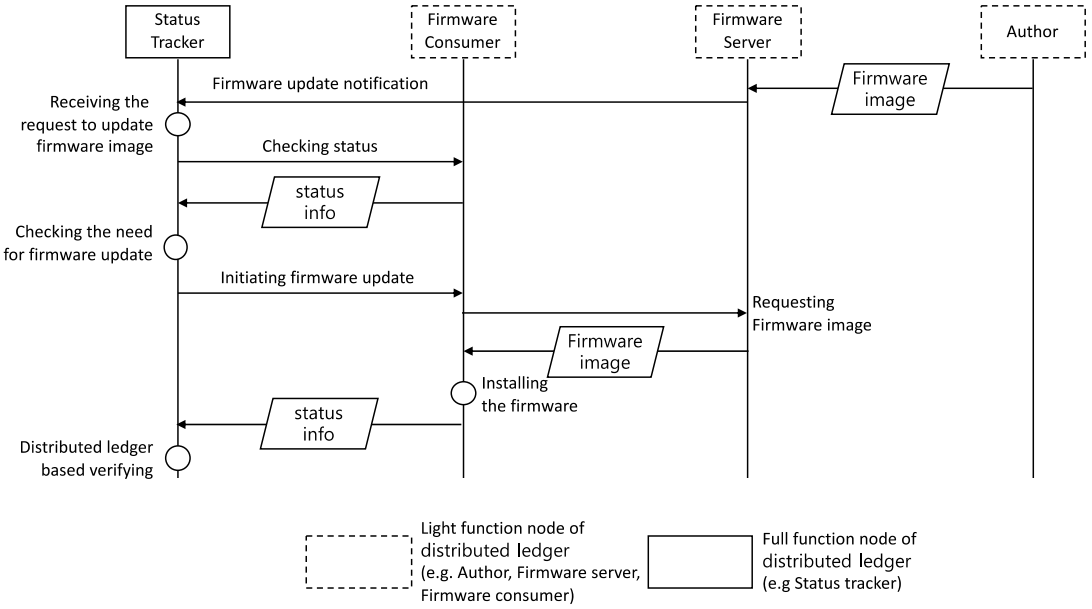

Figure II-1 – Distributed ledger based software update procedure

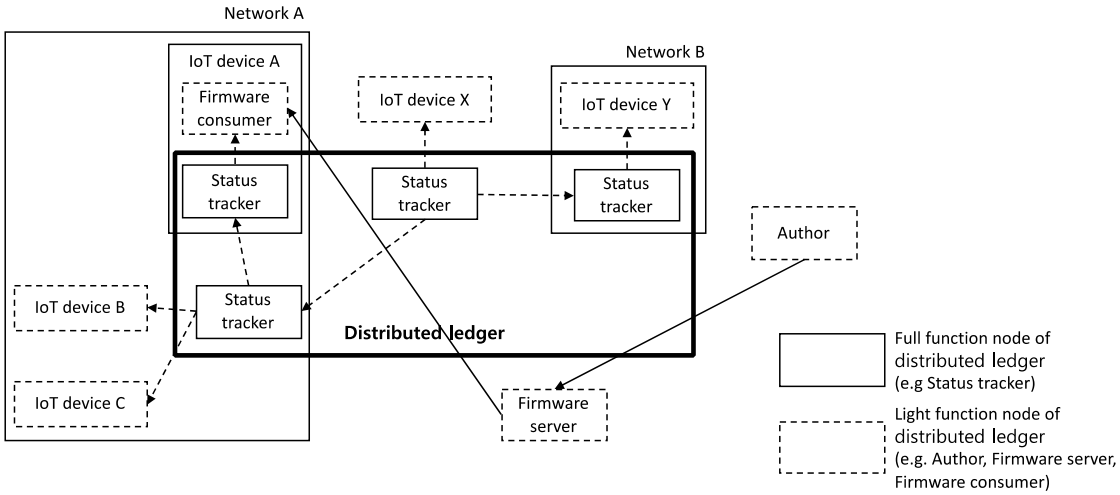

Figure II-2 – Distributed ledger based software update for multiple networks

Bibliography

[b-IETF manifest] Firmware Updates for Internet of Things Devices - An Information Model for Manifests, https://www.ietf.org/id/draft-ietf-suit-information-model-02.txt

[b-IETF architecture] A Firmware Update Architecture for Internet of Things Devices, https://www.ietf.org/archive/id/draft-moran-suit-architecture-03.txt

[b-IETF suit] IETF SUIT working group, https://datatracker.ietf.org/wg/suit/about/

[b-ISOC iotsu] IOTSU workshop, https://www.iab.org/activities/workshops/iotsu/

[b-oneM2M] Standards for M2M and the Internet of Things, http://www.onem2m.org/technical/published-drafts