ISSN(Online) : 2456-8910
International Journal of Innovative Research in Applied Sciences and Engineering (IJIRASE)
Volume 4, Issue 4, DOI: 10.29027/IJIRASE.v4.i4.2020.699-703, October 2020

Incorporating OMA LWM2M for Firmware Update

Murkal Shambunath Shankar[1], Dr. B.R.Shambhavi[2]
Department. of ISE, B.M.S College Of Engineering
Bengaluru, under VTU, Belagavi, India
Email: murkalshankar@gmail.com[1], shambhavibr.ise@bmsce.ac.in[2]

Abstract— Internet of Things (IoT) makes the Internet available to billions of users. Managing large quantities of devices is very critical. Among many other protocols, the global standards adopted Lightweight Machine 2 Machine(LWM2M) defines many traditional system management functions, such as remote device functions, firmware and software updates, communication monitoring, and security. Accessible Mobile Alliance Light-weight specification (OMA LwM2M) has given greater interoperability of applications to the absence of effort faced by the heterogeneous design, fragmented evolution environment and the interface between established solutions. This paper explains the firmware update architecture of LWM2M artefacts using standard OMA Specification. This involves downloading firmware kit, upgrading settings, upgrading states, and taking action after firmware updates

Keywords— Device Management, firmware update, object, OMA LWM2M.

I. INTRODUCTION

The Internet of Things (IoT) is the latest phenomenon in information technology and communications. It enables the connection of millions of devices over various networks to the Internet. Managing IoT / M2M devices is also a problem not just because of the large number of devices, but also because of their heterogeneity. The main aim is to enable both people and enterprise to be more productive, efficient and updated.

The Lightweight M2M (LWM2M) is an Open Mobile Alliance (OMA) framework standard. The main aim to build a well defined client-server quickly deployable to give system services. OMA LwM2M is a brand latest version developed specifically for devices with minimum memory, processing, and battery life durabilities, such as sensors, embedded devices, and wearables. These devices will be trending types of IoT / Machine2Machine devices; a significant IoT / M2M DM protocol will arise from the OMA LWM2M. The LWM2M specification provides device configuration APIs, connection monitoring/statistics, protection, firmware update, server supply, and so on. The commonly used CoAP (Constrained Application Protocol) provides LWM2M with in-built binding and thus makes the Internet of Things (IoT) especially attractive.

For IoT applications, Lightweight M2 M provides a range of core features, including secure bootstrapping, exploration of resources and frameworks, efficient management and application data transfer. The LWM2M server performs all interactions (read, write, etc.) by the specification, and where the client can start sending a notification for selected resources.

The main purpose of our model is how to update the firmware in LWM2M objects and see the responses on the client and server-side and also it explains about the Object definitions and Resource definitions with their operations and responses and also their states of mechanism for a firmware update.

This paper is divided into 6 sections. In the first section will give a brief introduction to the work. In the second section some Existing systems are studied. The third section will be about the model approaches. Fourth section will brief about performance steps with proposed system and in Fifth section results are analyzed. Sixth section will conclude about the proposed system. At the end some reference papers are presented.

II. RELATED WORKS

Z. Sheng et al[1] introduces an implementation of a low-cost IoT gateway inside an embedded device that was used in an IoT study of monitoring systems. It also addresses protocols for converting various sensor data into a standard format and provisioning methods that are supported by the server. This specifies the use of CoAP instead of HTTP as a communication protocol and offers multiprotocol gateway architecture. Also addressed is the M2M architecture which incorporates various other system management frameworks and M2M network management issues, it also covers IP-based network management protocols and ongoing self-management research.

Application management and remote control of IoT devices between LWM2M client and server are being addressed by S Datta and C Bonnet[2][3] where they explained about updates in machine 2 machine in modules.

A. Sehgal et.al[4] primarily describes a smart M2M gateway-based architecture designed to handle vast volumes of M2 M devices and incorporate the internal configuration of the gateway and M2M application and endpoint management APIs. They demonstrate a gateway in a smart home scenario which manages connected devices.

Wei-gang Chang and fuchun Joseph[5] resolved both problems by developing an OMA LWM2M gateway between OMA server and non-OMA LWM2M devices and incorporating the gateway as standard architecture into the ETSI M2M and oneM2M.

Suhas Rao, Devaiah Chendanda, Chetan Deshpande[6] discussed issues related to client-side design for LWM2M and its full deployment system performed over IoT nodes based on Contiki. They introduced Lightweight IoT protocol

stack, integrating the architecture of the LWM2M client machine with there interaction. And they have introduced real-world applications to test their usability and their effectiveness.

Abdulkadir Karaagac, Matthias Van Eeghem, Rossey, Bart Moons, Eli Poorter, Jeroen Hoebeke[7], suggest LwM2 M extensions to increase communications efficiency and add intermittent connectivity to enhance Low-Power Wide Area Network (LPWAN) support in LwM2M. For this purpose, we are introducing two new object models, namely Notify and Batch.

The Notify Object allows reverse interaction models and Ready-to-Receive (RTR) functionality to be developed, enabling LwM2 M clients to send periodic resource updates without requiring a request, and to notify LwM2 M servers that they are ready to receive downlink messages and to keep network connectivity open when downlink communication is needed. Finally, the Batch object enables LwM2 M servers to perform actions on multiple resources within a system by using a single request.

David Tracey [8] describes an architecture that uses a tuple-space-based library for data flow from sensors to applications with specified service abstractions. This also contrasts the LWM2M Information Model for OMA and the Standard Information Model for DMTF. It introduces a 'C' implementation of the OMA LWM2M model on our Contiki3.0 OS tuple space and takes into account the efficiency of our architecture and its alignment with existing CoAP and OMA LWM2M implementations.

III. Model Approaches

This enabler characterizes the communication agreement between an LwM2M server and LwM2M client in an LwM2M system. The OMA Light-weight M2M empowering agent for LwM2M Devices incorporates the executive's gadget and administration enabling. For this empowering agent, the target LwM2M devices are fundamentally asset-obliging gadgets.

This authorizing agent, therefore, uses light and reduced convention just as an effective model of asset knowledge. For the LwM2M Enabler, a client-server configuration is introduced, where the LwM2M interface operates as an LwM2M Client, and the M2M administration, stage or application operates as the LwM2M Server. There are two parts on the LwM2M Enabler, LwM2M Server and LwM2M Client.

Between LwM2M Client and LwM2M Server sections four interfaces are designed as shown as follows

- Bootstrapping
- Client Registration
- Device and service enablement
- Information Reporting

Where Figure1 tells us, for communication between Client and Server, LWM2M Enabler uses both the UDP Constrained Application Protocol (CoAP) and the SMS bonds between client and server. The Transport Layer Security (DTLS) datagram gives User Datagram Protocol transport layer more stability to interact with each other. The protocol stack from the LwM2M Enabler is shown in Figure2 which show their relation between transport and messaging protocol.

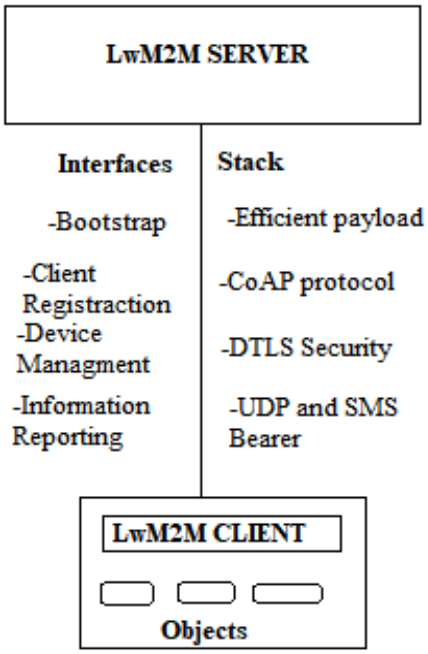

Figure 1: Architecture of LwM2M Enabler

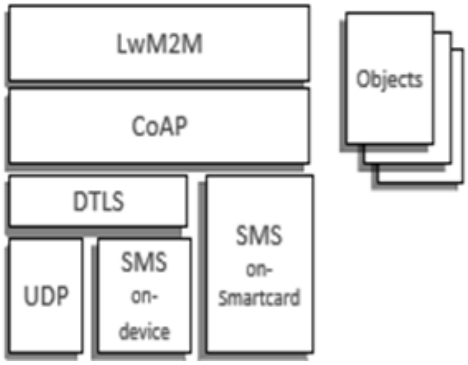

Figure 2: The protocol stack for LwM2M Enabler.

Table 1: List of objects defined in LWM2M

0	Security
1	Server
2	Access Control
3	Device Information
4	Connectivity Monitoring
5	Firmware Version
6	Location Details
7	Connectivity Statistic

IV. PROPOSED SYSTEM

Figure 3 displays a potential implementation of the UML 2.0 state diagram for the firmware update process. Where figure consists of states where represented in rectangle and arrows connecting each state to get firmware updated and each state must be completed.

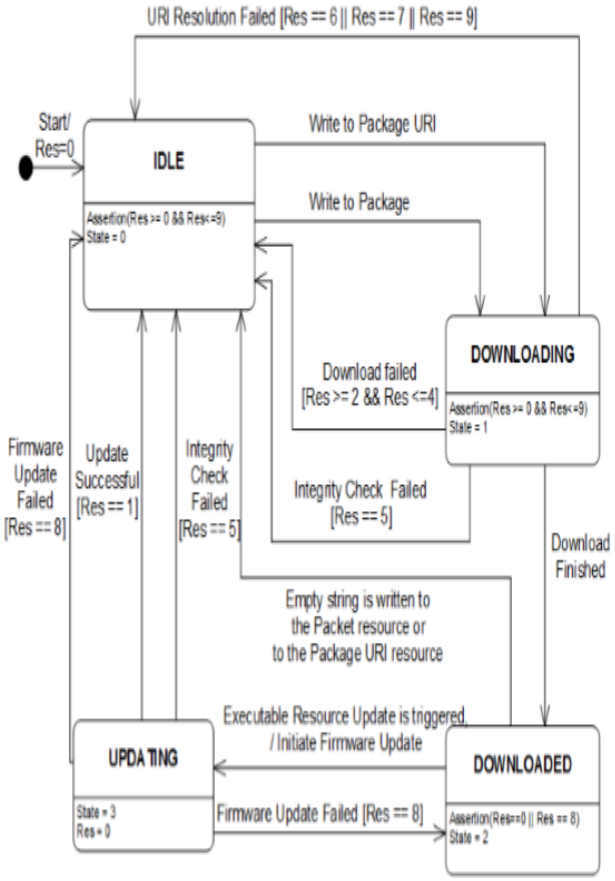

Figure 3: Firmware update mechanisms

The state diagram consists of four states namely:

STATE 0: Where State 0 is in an IDLE state as shown in the figure and to start an event trigger must be done using resource 0. The assertion of resource from 0 to 9 is represented in this state wherefrom State-0 to State-1 package URI is written for the successful condition.

STATE 1: State-1 is in DOWNLOADING state when State-1 gets URI from State-0 as shown in figure 3. The assertion of resource from 0 to 9 is represented in this state. When an event is successful than it shows as the download finished for successful condition and for failed condition notification goes from State-0 to State-1 as URI resolution failed, download failed with their resource number.

STATE 2: State-2 is in the DOWNLOADED state when State-2 gets downloaded message from State-1 as shown in figure 3. The assertion of resource from 0 to 8 is represented in this state. When an event is successful than it initiates firmware update to State-3 in successful condition and for failed condition notification goes from State-2 to State-1 as an empty string is written to packet resource their resource number.

STATE 3: State-3 is in UPDATING state when State-3 gets initiate firmware update message from State-2 as shown in figure 3. The assertion of resource 0 is represented in this state. When an event is successful than it initiates firmware update is successful to State-1 in successful condition and for failed condition notification goes from State-3 to State-1 as firmware update failed with their resource number.

This LWM2M thing enables the control of the modified firmware. This thing involves installing the firmware kit, upgrading firmware, and executing behaviour after firmware upgrading. The firmware update can allow the computer to reboot and it depends on a variety of variables, such as the configuration of the operating system and the frequency of the software updates.

By using DTLS security assurance LWM2M adaptation 1.0 will give access to LWM2M client to communicate with LWM2M server with their latest structure of firmware version. There are many devices which should be updated we can update by adding their delta files using firmware over the air to all devices. Models for such plan choices are how to deal with the firmware version storehouse at the server side (which may incorporate UI contemplations), the procedures to give extra application layer security insurance of the firmware picture, what several variants of firmware envision to save on the device, and how to execute the firmware update process considering the equipment which is suitable for an IoT equipment item.

The perspective is viewed as outside the extent of the LWM2M form 1.0 determination. An LwM2M Server may likewise teach an LwM2M Client to bring a firmware picture from a committed server (rather than pushing firmware envisions to the LwM2M Client). The Package URI asset is contained in the Firmware protest and can be utilized for this reason. An LwM2M Client MUST help square astute exchange [CoAP Block wise] on the off chance that it actualizes the Firmware version thing. An LwM2M Server must help square astute exchange. Different conventions, for example, HTTP/HTTPs, MAY likewise be utilized for downloading firmware refreshes (using the Package URI asset). For obliged gadgets, it is, in any case, RECOMMENDED to utilize CoAP for firmware downloads to maintain a strategic distance from the requirement for extra convention executions.

Object definition

Name	Object ID	Instances	Mandatory	Object URN
Firmware Update	5	Single	Optional	urn:oma:lwm2m:oma:5

Figure 4: Example of the object definition.

Object Definition: As shown in figure 4 proper Firmware update name should be assigned and Object ID must be declared with there Instances and Object URN for the update.

Resource Definition: Following names should be defined with there operations, type and range

a. Package: Firmware package name should be defined with the write operation.

b. Uniform Resource Identifier: Alternative Uniform Resource Identifier firmware packet is used for Read-Write operation for which system will access single string 0-255bytes. After receiving the Package URI the system will conduct the update at the next practicable opportunity. RFC 3986 determines URI format. The protocol to be used is determined by the URI scheme. This endpoint will be an LwM2M Server for CoAP but does not necessarily have to be. A CoAP server implementing block-wise transfer is adequate as a server hosting a firmware repository and it is anticipated that this server can only act as a separate firmware file server.

c. Update: Perform operation and software updates by using the software delta file in the Package, or by using the firmware downloaded from the Package URI. This function can only be executed when the state resource is downloaded.

d. State: Perform Read operation of Integer type Specifies the latest state of this firmware update. The LwM2M client sets this value.

1. Idle (before downloading, or after updating successfully)
2. Download (The series of data is underway)
3. Click here to download
4. Updating with new delta file if it is successful.

If the device has downloaded the firmware packages from the Package URI the Downloaded state changes. Restores the Firmware Update State Machine by writing an empty string to Package Resource or Package URI Resource: the State Resource value is set to Idle and the Update Resource value is set to 0. When triggering the executable Resource Update in Downloaded State, the state will shift to Update. If delta file is updated than state goes to idle with the new version of firmware or else it goes back to state 2 were shown in figure3.

e. Update Result: Perform Read operation of type Integer, Contains the result of a firmware download or upgrade. The integer read operation is done by LWM2M client. Where each of them is explained from 1 to this happens when uniform resources are unsupportive file, by seeing those resource numbers we come to know which type of error occurred and this can be corrected

1. Initiation of the updating process.
2. The system successfully updated.
3. Lack of space for the new delta file.
4. RAM out of download state.
5. Missed connection during download state.
6. Credibility checks failure of new kit downloaded.
7. File not supported.
8. Permitted Uniform Resource.
9. New delta files not working.
10. Unassisted file.

f. Package name: Perform Read operation of type String, Name of the Firmware Package.

g. Package Version: Performs String style read process, Firmware module edition.

h. Firmware Update Protocol Support: LWM2M client send firmware delta file which performs Integer type read operation. The LwM2M server uses this information to access which URI should be included in the Package URI. An LWM2M server does not include a uniform resource in the Package URI object using a protocol that is not accepted by the LWM2M client.

0	CoAP support block-wise transfer.
1	CoAP supports extra delta file transmission.
2	HTTP1.1
3	HTTPS1.1

i. Firmware Update Delivery Method: The LwM2M Client uses this tool to demonstrate so send firmware delta packets to devices using push and pull methods where this URI resource are shown in the table

0	PULL
1	PUSH
2	BOTH PUSH and PULL

V. RESULTS AND ANALYSIS

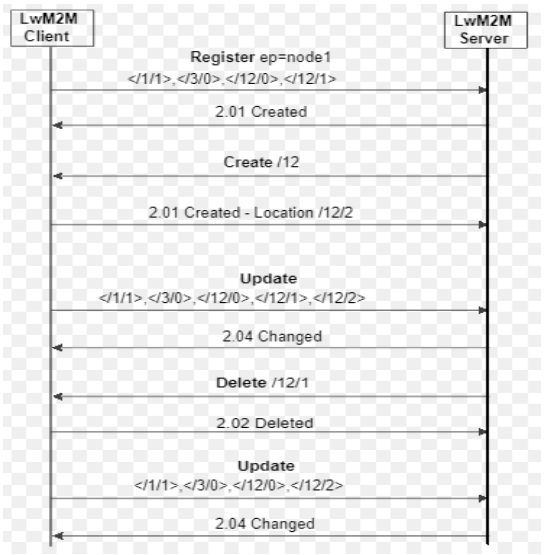

Figure 5: Updated firmware.

This section will describe the results obtained by using proposed system based on the stages. If the stages are clear then the firmware update is being done. Where we can see in figure 5, client is registered with server to update with new firmware version where delta file must be added to the existing version. When delta file is added by passing through each state from idle to updating we can see in figure 5 where 2.01 is updated to 2.04.

VI. Conclusion

In this paper, I explained detail about Firmware update in LWM2M objects with their stages of object definition and resource definition. Delta file is a combination of the new file which are newly added, where this delta file must be added to the old version to get new version this process know as Firmware update. This process is successful when we pass through each state form 0 to 3 and where we get Integer resource number if any step had been failed so that we can directly correct at that step by seeing error code. The firmware update improves the performance and optimization are being done. The firmware update makes it more efficient, leading to increased performance and speed of IoT modules.

References

[1] Sheng, Z., Wang, H., Yin, C., Hu, X., Yang, S., & Leung, V. C. (2015). Lightweight management of resource-constrained sensor devices in internet of things. *IEEE internet of things journal*, *2*(5), 402-411.

[2] Datta, S. K., & Bonnet, C. (2015, April). A lightweight framework for efficient M2M device management in oneM2M architecture. In *2015 International Conference on Recent Advances in Internet of Things (RIoT)* (pp. 1-6). IEEE.

[3] Datta, S. K., & Bonnet, C. (2014, September). Smart m2m gateway based architecture for m2m device and endpoint management. In *2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)* (pp. 61-68). IEEE.

[4] Sehgal, A., Perelman, V., Kuryla, S., & Schonwalder, J. (2012). Management of resource constrained devices in the internet of things. *IEEE Communications Magazine*, *50*(12), 144-149.

[5] Chang, W. G., & Lin, F. J. (2016, October). Challenges of incorporating OMA LWM2M gateway in M2M standard architecture. In *2016 IEEE Conference on Standards for Communications and Networking (CSCN)* (pp. 1-6). IEEE.

[6] Rao, S., Chendanda, D., Deshpande, C., & Lakkundi, V. (2015, August). Implementing LWM2M in constrained IoT devices. In *2015 IEEE Conference on Wireless Sensors (ICWiSe)* (pp. 52-57). IEEE.

[7] Karaagac, A., VanEeghem, M., Rossey, J., Moons, B., DePoorter, E., & Hoebeke, J. (2018, October). Extensions to LwM2M for intermittent connectivity and improved efficiency. In *2018 IEEE Conference on Standards for Communications and Networking (CSCN)* (pp. 1-6). IEEE.

[8] Tracey, D., & Sreenan, C. (2017, May). OMA LWM2M in a holistic architecture for the Internet of Things. In *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)* (pp. 198-203). IEEE.

[9] Open Mobile Alliance (OMA), "LightweightM2M Technical specification v1.0", available online at www.openmobilealliance.org.