LwM2M TITLE WIP

Jaime Jiménez
jaime.jimenez@ericsson.com
Ericsson
Jorvas, Finland

ABSTRACT

This paper provides an overview of the Lightweight Machine-to-Machine (LwM2M) protocol, highlighting its role in IoT device management and its integration with key IETF standards such as CoAP, DTLS, and OSCORE. While network management protocols like SNMP, RESTCONF, and CORECONF focus on monitoring and configuring endpoints within the network, LwM2M extends these capabilities by managing the entire lifecycle of individual devices, including configuration, control, and maintenance. We discuss LwM2M's architecture, data model, communication interfaces, and explore recent advancements in the protocol.

KEYWORDS

Internet of Things, IoT, network management, device management, security, standards

Reference:

Jaime Jiménez. 2024. LwM2M TITLE WIP. In *submissions to the IAB Next Era of Network Management Workshop*, 5 pages.

Internet Architecture Board (IAB) Workshop, December 2024,
.

1 INTRODUCTION

The rapid evolution of network management protocols necessitates a reevaluation of existing technologies and their applicability to modern challenges. The Lightweight Machine-to-Machine (LwM2M) protocol [3], developed by the Open Mobile Alliance (OMA), is a key player in this domain, offering a standardized framework for managing Internet of Things (IoT) devices [4]. This paper explores the role of LwM2M in the context of the IAB workshop on the Next Era of Network Management Operations, focusing on its current deployments, challenges, and future potential.

The IAB "NEMOPS" workshop seeks contributions that critically assess the progress made since the 2002 IAB workshop, particularly in terms of network management protocols. This paper aims to present LwM2M, an management protocol that addresses the needs for managing IoT endpoints from the operational point of view of device and network management.

Our contribution is informed by the authors' extensive experience with IoT and contributions in the IoT domain both at IETF and in OMA. The rest of the document is organized as follows: The Introduction outlines LwM2M as a standardized framework for managing IoT devices, addressing current network management challenges. The LwM2M Protocol Overview details its architecture, focusing on communication between Clients, Servers, and Bootstrap Servers (see Figure 1). Recent advancements, including integrations with blockchain and industrial protocols, are discussed in LwM2M Extensions, highlighting its adaptability. Remaining adoption challenges are explored in LwM2M Operations, while the Conclusions summarize key findings and future directions.

LwM2M and IETF

The IETF has played a fundamental role in shaping the protocols that underpin LwM2M. IETF efforts have

focused on adapting existing Internet and Web protocols to meet the needs of resource-constrained IoT devices [18].

LwM2M is built upon several key IETF standards. At the core is the *Constrained Application Protocol (CoAP)* [27], a lightweight RESTful protocol designed for constrained environments, providing the fundamental request/response model for LwM2M communications. *RFC 7959* [8] defines block-wise transfers in CoAP, allowing LwM2M to efficiently handle large payloads by breaking them into smaller blocks. The *LwM2M* protocol also leverages *RFC 7641* [13] for resource observation, enabling clients to subscribe to changes of resource state without continuous polling. For secure communications, LwM2M often relies on the *Datagram Transport Layer Security (DTLS)* as outlined in *RFC 6347* [24], ensuring encryption and integrity over the CoAP protocol. An additional layer of security is provided by *Object Security for Constrained RESTful Environments (OSCORE)* [25], which offers end-to-end encryption and integrity protection directly at the application layer, making it suitable for scenarios where DTLS is not applicable. Additionally, the *Constrained RESTful Environments (CoRE) Resource Directory (RD)* [5] facilitates resource registration of IoT endpoints by maintaining information about resources on other servers, the lookup interface, although present is not intended for applications but for management purposes.

Furthermore, LwM2M supports additional transport protocols, including the *Hypertext Transfer Protocol (HTTP)* [12] and *Message Queuing Telemetry Transport (MQTT)* [21], the latter developed by the OASIS foundation [2]. This broadens its applicability across various network environments and use cases.

2 LWM2M PROTOCOL OVERVIEW

This section presents an overview of the LwM2M protocol, emphasizing its participating entities, data model, and communication interfaces. Additionally, it describes the LwM2M library's design and functionality, highlighting challenges encountered during its development.

Participating Entities

The LwM2M protocol defines three primary entities that form the backbone of its communication architecture:

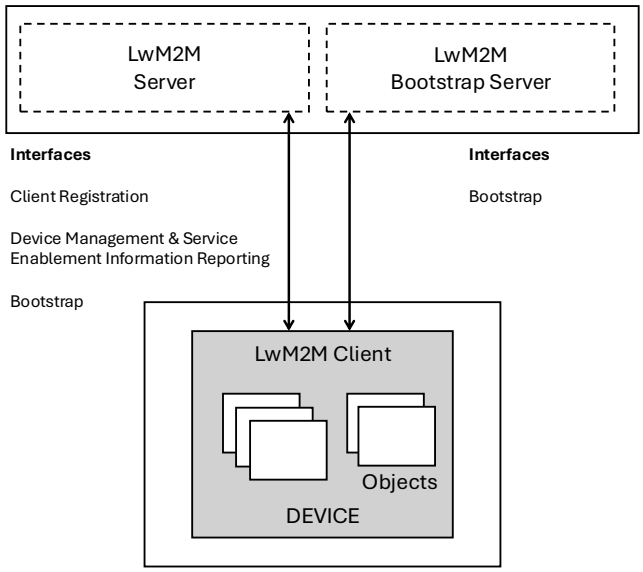

Figure 1: General LwM2M Architecture

- **LwM2M Client**: Typically a smart device, such as a light, smart meter, switch, or gateway, responsible for data collection and resource management at the device level. The client oversees the lifecycle of various objects and communicates with servers to transmit data and receive management instructions.
- **LwM2M Server**: Often referred to as the manager, this server manages multiple clients, aggregates their data, and issues configuration or maintenance commands. It is essential for controlling and coordinating client devices within an IoT ecosystem.
- **LwM2M Bootstrap Server**: This entity is responsible for the initial configuration of the LwM2M Client. Unlike other protocols, LwM2M includes a Bootstrap Server to streamline device setup, particularly when multiple servers are involved. During the bootstrap process, it provides the client with configuration details, such as security credentials and connection information, which can be pre-integrated into the device's software or dynamically supplied by the Bootstrap Server.

The introduction of the Bootstrap Server differentiates LwM2M from other IoT protocols. Before the client can

establish a connection to a server, it undergoes the bootstrap procedure to load initial configurations and key material. This feature is particularly advantageous in scenarios with multiple servers or when load balancing is required, as it allows for flexible and dynamic configuration.

LwM2M Data Model

The LwM2M protocol employs a structured, object-based data model to facilitate communication between clients and servers. In this model, each data entity is defined as an object, uniquely identified by an integer ID, as specified by the Open Mobile Alliance (OMA). These objects can represent various resources, such as sensors, actuators, or configuration settings.

Each object consists of multiple resources, which serve as the fundamental data points within the object. Resources are assigned integer IDs and are classified as either mandatory or optional, depending on the object's intended function being the mandatory ones used for interoperability purposes. An object can also have multiple instances, each with a unique instance ID, allowing for flexibility. For instance, a device with several sensors can represent each sensor as an instance of the same object, such as in a network of connected lightbulbs.

The LwM2M data model enables servers to access individual resources, instances, or entire objects using well-defined URI strings. The URI template approach follows the Web Linking and the IETF CoRE Link Format [26]. The format is as follows:

$$/< ObjectID > /< InstanceID > /< ResourceID >$$

In this structure, the instance or resource ID can be omitted if the request targets the entire object or a specific instance. This model provides a straightforward way to interact with data at different levels of granularity.

All objects are listed in the LwM2M Registry [23], categorized as either application-oriented (e.g., thermostats, lamps) or system-oriented (e.g., security, connectivity). Those for applications follow the same schema and come from the IP for Smart Objects (IPSO) [14] consortium which was integrated into OMA.

Semantically, the object type represents a single measurement, actuation, or control point for example a temperature sensor, a light (actuator), or an on-off switch (control point). A resource specifies a particular view or active property of an object. For example, a temperature sensor object might expose the current value (most recent reading), also the minimum and maximum possible reading, the minimum and maximum reading in an interval, and attributes like engineering units and application type. Attributes describe the metadata configuration, settings, and state of an object or resource, and are discoverable by reading the link-format data of an object or resource. Multiple attributes may be serialized in the link-format descriptors that an object exposes.

Communication Interfaces

LwM2M defines four primary communication interfaces that facilitate interactions between clients and servers, each serving distinct roles in the protocol's operation:

- **Bootstrap Interface**: This interface facilitates the initial setup by enabling the client to obtain necessary configurations and security credentials from the Bootstrap Server prior to connecting with an LwM2M Server. The process can be either automated or client-initiated.
- **Registration Interface**: After the bootstrap procedure, the client registers with the LwM2M Server through the Registration Interface. During registration, the client provides its endpoint name, which acts as an access token. The server can deny access if the provided token is invalid. To maintain its registration status, the client must periodically send updates. If these updates are not received within the agreed time, the client is considered de-registered and must re-initiate the registration process.
- **Device Management and Service Enablement Interface**: Once registered, the server can use this interface to perform various management tasks, such as reading and writing data or executing commands on the client. It allows the server to control the client's resources, adjusting settings or triggering actions as required.
- **Information Reporting Interface**: This interface enables the client to report changes in its status or resource values to the server, using the Observe

function. The server can set up observation requests, prompting the client to notify it when certain resource values change or when predefined conditions are met. This capability is particularly useful for monitoring dynamic IoT environments where timely updates are crucial. LwM2M defines specific attributes to configure how frequently these notifications occur.

LwM2M has been integrated on most IoT OSs like Mbed [1], RIOT OS [7], Contiki-NG [10], and many others, with a strong focus on IoT device security [11].

3 LWM2M EVOLUTION

LwM2M, though robust in its core functionalities, has advanced significantly through various extensions and integrations. This section explores recent research developments that may inspire its future evolution. LwM2M has become a central focus in IoT research due to its improved security, in particular for firmware updates [30] and integration with other technologies.

Several studies have concentrated on enhancing IoT security through the use of LwM2M. For instance, Muhammad et al. [19] investigated the use of ARIA cryptography within Hardware Secure Modules for both LwM2M and MQTT protocols, aiming to bolster the security of next-generation IoT systems. Similarly, Lanzieri et al. [17] proposed extensions to the LwM2M core specification to facilitate secure and authorized client-to-client communication, thereby addressing a significant limitation of the current standard.

Others have focused on interoperability between SDOs, Kim et al. [16] have designed and implemented a blockchain-based system that enables interworking between oneM2M [22] and LwM2M IoT systems. This approach not only improves interoperability between different IoT frameworks but also enhances security by leveraging blockchain's immutable and distributed ledger properties.

In the context of Industrial IoT (IIoT), Yaker et al. [29] introduced a novel edge Security Information and Event Management (SIEM) system for managing IoT flows within 5G private networks. Their approach incorporates LwM2M data events, demonstrating how edge computing can effectively manage and secure IoT data in a 5G environment. Similarly, Myoung et al. [20] addressed the integration of LwM2M with smart metering technologies, proposing a data interworking model between the Device Language Message Specification (DLMS) [6] used for managing smart meters and the LwM2M protocol. This highlights the potential for harmonizing different IoT protocols within metering infrastructure systems.

Further efforts have been made to integrate LwM2M with industrial communication protocols. Karaagac et al. [15] explored the interoperability between LwM2M and the Open Platform Communications Unified Architecture (OPC UA), proposing a framework where OPC UA Servers can be virtualized as LwM2M Clients and vice versa. This approach aims to bridge the gap between IoT and industrial automation, facilitating seamless communication across heterogeneous systems. Similarly, Cavalcanti et al. [9] reviewed various machine-to-machine communication protocols within the context of Industry 4.0, emphasizing the advantages of integrating OPC UA with LwM2M for industrial applications.

A last example, Wang et al. [28] proposed a CoAP-based OPC UA transmission scheme tailored for resource-constrained devices. Although their work does not directly involve LwM2M, it illustrates the broader trend of adapting industrial communication protocols to fit the needs of IoT environments and the reuse of IETF protocols in wider contexts that originally intended. This could also indicate the interest in enabling resource-efficient and interoperable solutions in the IoT landscape, where industrial and IoT domains converge.

4 CONCLUSIONS

In this paper, we have presented an overview of LwM2M and its components, the dependencies on IETF standards and some potential new building blocks for future additions.

5 ACKNOWLEDGMENTS

We'd like to thank Ericsson for their support of this work.

REFERENCES

[1] [n. d.]. *Mbed OS.* https://os.mbed.com/ Accessed: 2023-10-10.
[2] [n. d.]. OASIS Message Queuing Telemetry Transport (MQTT) Technical Committee. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt.

[3] Open Mobile Alliance. 2018. *Lightweight Machine to Machine Technical Specification.* https://www.openmobilealliance.org/release/LwM2M/V1_0-20170208-A/OMA-TS-LightweightM2M-V1_0-20170208-A.pdf

[4] Open Mobile Alliance. 2023. *Open Mobile Alliance Standards Development Organization.* https://www.openmobilealliance.org/ Accessed: 2023-10-10.

[5] C. Amsüss, Z. Shelby, M. Koster, C. Bormann, and P. van der Stok. 2022. Constrained RESTful Environments (CoRE) Resource Directory. RFC 9176. https://datatracker.ietf.org/doc/html/rfc9176 Standards Track.

[6] DLMS User Association. 2020. *DLMS/COSEM: The Global Standard for Smart Metering and Energy Management.* DLMS User Association, Geneva, Switzerland. https://www.dlms.com/.

[7] Emmanuel Baccelli, Oliver Hahm, Mesut Gunes, Matthias Wahlisch, and Thomas C. Schmidt. 2015. RIOT OS: Towards an OS for the Internet of Things. *Proc. IEEE* 103, 4 (2015), 1–14.

[8] C. Bormann and Z. Shelby. 2016. Block-Wise Transfers in the Constrained Application Protocol (CoAP). https://www.rfc-editor.org/rfc/rfc7959.html

[9] Marcella Cavalcanti, Hugo Costelha, and Carlos Neves. 2023. Industry 4.0 Machine-to-Machine Communication Protocols and Architectures on the Shop Floor. (2023), 222–234.

[10] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. 2004. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks* (2004), 455–462.

[11] J Ellamathy. 2023. *Securing LwM2M with Mbed TLS in Contiki-NG.* diva-portal.org. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1751815 Query date: 2024-10-08 22:53:23.

[12] R. Fielding and J. Reschke. 2014. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. https://doi.org/10.17487/RFC7230

[13] K. Hartke. 2015. Observing Resources in the Constrained Application Protocol (CoAP). https://www.rfc-editor.org/rfc/rfc7641.html

[14] J. Jimenez, M. Koster, and H. Tschofenig. 2016. IPSO Smart Objects. Position paper for the IOT Semantic Interoperability Workshop. https://www.ietf.org/ietf-ftp/slides/slides-iotsiws-ipso-smart-objects-00.pdf Accessed: 2024-10-16.

[15] Abdulkadir Karaagac, Niels Verbeeck, and Jeroen Hoebeke. 2019. The Integration of LwM2M and OPC UA: An Interoperability Approach for Industrial IoT. (2019), 313–318. https://doi.org/10.1109/WF-IoT.2019.8767209

[16] Donggyu Kim, Uk Jo, Yohan Kim, Yustus Eko Eko, and Howon Kim. 2023. Design and implementation of a blockchain based interworking of oneM2M and LWM2M IoT systems. *Journal of Information Processing Systems* 19, 1 (2023), 89–97.

[17] Leandro Lanzieri, Peter Kietzmann, Thomas C. Schmidt, and Matthias Wählisch. 2022. Secure and Authorized Client-to-Client Communication for LwM2M. (2022), 158–170. https://doi.org/10.1109/IPSN54338.2022.00020

[18] Roberto Morabito and Jaime Jimenez. 2020. IETF Protocol Suite for the Internet of Things: Overview and Recent Advancements. *IEEE Communications Standards Magazine* 4, 2 (2020), 41–49. https://doi.org/10.1109/MCOMSTD.001.1900014

[19] I Muhammad, LAM Ari, and D Pratama. 2024. Next-Gen IoT Security: ARIA Cryptography within Hardware Secure Modules—A Comparative Analysis of MQTT and LwM2M Integration. *Proceedings of the Korea Society for Internet Information* (2024). https://koreascience.kr/article/CFKO202422572150314.page Query date: 2024-10-08.

[20] Nogil Myoung, Yoojin Kwon, Myunghye Park, and Changsoo Eun. 2023. Data Interworking Model and Analysis for Harmonization of Smart Metering Protocols in IoT-Based AMI System. *Sensors* 23, 6 (2023). https://doi.org/10.3390/s23062903

[21] OASIS. 2014. MQTT Version 3.1.1. https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[22] oneM2M. 2023. oneM2M Technical Specification. https://www.onem2m.org/technical/onem2m-specifications.

[23] Open Mobile Alliance. [n. d.]. LwM2M Registry. https://github.com/OpenMobileAlliance/lwm2m-registry.

[24] E. Rescorla and N. Modadugu. 2012. Datagram Transport Layer Security Version 1.2. https://www.rfc-editor.org/rfc/rfc6347.html

[25] J. Selander, D. Palombini, F. Armknecht, G. Selander, and L. Seitz. 2019. Object Security for Constrained RESTful Environments (OSCORE). https://doi.org/10.17487/RFC8613

[26] Zach Shelby. 2012. Constrained RESTful Environments (CoRE) Link Format. RFC 6690. https://datatracker.ietf.org/doc/html/rfc6690

[27] Z. Shelby, K. Hartke, and C. Bormann. 2014. The Constrained Application Protocol (CoAP). https://www.rfc-editor.org/rfc/rfc7252.html

[28] Yi Wang, Chenggen Pu, Ping Wang, and Junrui Wu. 2020. A CoAP-based OPC UA Transmission Scheme for Resource-Constrained Devices. (2020), 6089–6093. https://doi.org/10.1109/CAC51589.2020.9326995

[29] K Yaker, BA Salem, B Pierard, et al. 2024. A Novel EDGE SIEM for Industrial IoT Flows Within 5G Private Networks. *2024 Global ...* (2024). https://ieeexplore.ieee.org/abstract/document/10449912/ Query date: 2024-10-08 22:53:23.

[30] Koen Zandberg, Kaspar Schleiser, Francisco Acosta, Hannes Tschofenig, and Emmanuel Baccelli. 2019. Secure Firmware Updates for Constrained IoT Devices Using Open Standards: A Reality Check. *IEEE Access* 7 (2019), 71907–71920. https://doi.org/10.1109/ACCESS.2019.2919760