machnation

Comparing the efficiency of LwM2M and MQTT: hands-on test results of two technology clients on a typical IoT device

April 2020

Table of Contents

Table of Contents 2
Executive Summary 3
Introduction 5
Methodology and Technical Specifications 6
    Device Clients 7
    Security 8
    Packet Capture 8
    Data Observations 8
Test Findings 9
    Packet analysis: initial connection 10
    Packet analysis: steady-state without observation reporting 12
    Packet Analysis: observation reporting at 30-second reporting intervals 14
    Packet analysis: single platform-to-device message 17
    Packet analysis: OTA firmware update 19
    Power consumption: idle, 1, 30, and 60-second observation reporting intervals 22
    Qualitative evaluation: business comparison 24
    Qualitative evaluation: technical comparison 24
Areas of Future Suggested Research 26
Conclusions 27
About Project Sponsors 28
    About AT&T Communications 28
    About AVSystem 28
    About MachNation 28

Executive Summary

When enterprises and their developers begin to select technology components for an Internet of Things (IoT) deployment, they are confronted with many choices. These choices span the IoT technology stack including devices, protocols, communication services, communications management, various types of IoT platforms, and application development.

One of the first choices that developers make is selecting software and protocols that allow devices to communicate to IoT platforms and, in some cases, allow cloud platforms to manage IoT devices. Two of the most common protocols to perform these tasks are Lightweight Machine-to-Machine (LwM2M) and Message Queuing Telemetry Transport (MQTT).

Today, MQTT is the communications protocol chosen by many enterprises deploying IoT solutions. There is a fairly well-developed ecosystem of vendors that support and market MQTT, offer productized MQTT clients, and supply documentation to support an enterprise's MQTT implementation.

LwM2M, by contrast, is a newer technology solution offering a communications and device management protocol. While the potential exists for LwM2M to be heavily adopted by enterprise developers, today the LwM2M ecosystem is not as well developed as the ecosystem for MQTT and it remains more difficult for an enterprise developer to rapidly design an IoT solution around LwM2M compared to MQTT[1]. This situation must be changed if LwM2M and supporting vendors wish to incent developers to design more IoT solutions using LwM2M.

By choosing the right technology protocol, IoT developers will help enterprises bring their IoT solutions to market faster; save ongoing development, management, and operations costs including communications services expenses; and future-proof their IoT solutions.

In order to help enterprises and developers understand the differences between the Open Mobile Alliance (OMA) LwM2M protocol and the more traditional MQTT messaging protocol, MachNation designed and completed a set of tests on a typical IoT device using an MQTT client and a LwM2M client. With technology support from AT&T and AVSystem, MachNation strove to create a set of tests that simulates a meaningful, real-world IoT deployment.

Based on the testing, MachNation arrived at a set of conclusions (see Figure 1). In particular,

LwM2M shows efficiency and performance benefits over MQTT in the following:

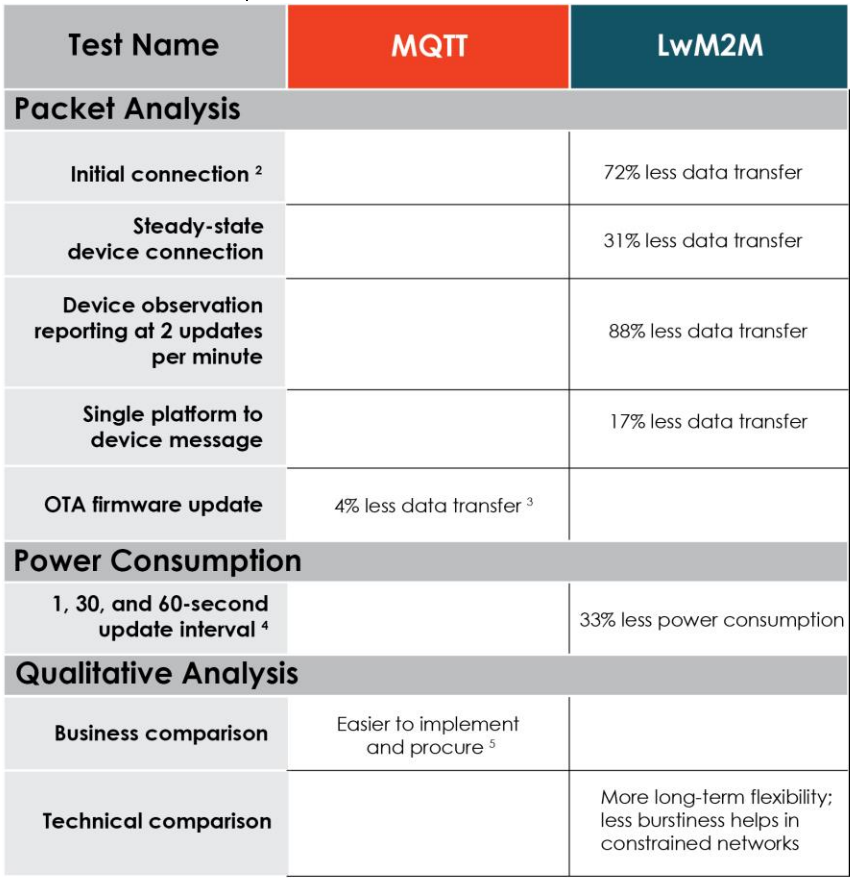
- Less data transfer during the initial device-to-platform connection and after a device reboot
- Less data transfer during the ongoing steady state of a device connection
- Less data transfer during device observations at 2 updates per minute
- Less data transfer during a single platform-to-device message push
- Less bursty, allowing better network planning, improved efficiency, and possibly lower operations costs associated with communication services in constrained networks
- Less power consumption than a similarly-equipped MQTT device irrespective of update interval, although it is possible that such differences stem from the SDK language rather than the protocol used
- Yields important, long-term technical advantages, but more forethought required for implementation during the design and development process.

MQTT shows efficiency and performance benefits over LwM2M in the following:

- Less data transfer during an over-the-air (OTA) firmware update
- Easier and faster to deploy on an IoT device, though leaves unresolved many important device-integration questions during initial deployment and rollout, possibly creating unknown, future costs for enterprises.

[1] The relative difficulties in implementing LwM2M could arise from technology, documentation, or vendor-support issues. However, it is quite possible that as LwM2M matures, these deficiencies will be addressed.

FIGURE 1: SUMMARY OF EFFICIENCY AND PERFORMANCE SCORES FOR LWM2M AND MQTT CLIENTS ON A TYPICAL IOT DEVICE [SOURCE: MACHNATION, 2020]



Test Name	MQTT	LwM2M
Packet Analysis		
Initial connection [2]		72% less data transfer
Steady-state device connection		31% less data transfer
Device observation reporting at 2 updates per minute		88% less data transfer
Single platform to device message		17% less data transfer
OTA firmware update	4% less data transfer [3]	
Power Consumption		
1, 30, and 60-second update interval [4]		33% less power consumption
Qualitative Analysis		
Business comparison	Easier to implement and procure [5]	
Technical comparison		More long-term flexibility; less burstiness helps in constrained networks

Based on the testing completed, MachNation finds that a LwM2M client is preferable to an MQTT client for devices that:

- Are designed for long battery life, including those that support IoT applications where device replacement is difficult and costly
- Operate on a constrained network (e.g., LTE Cat-M1 and NB-IoT) or where network traffic is metered
- Remain in a dormant state relatively often.

MachNation also recommends that embedded developers implementing a LwM2M-powered solution seek a fairly productized (rather than open source) LwM2M client upon which to build. Vendors that offer productized clients will generally offer technical support to aid enterprises in deployment. This can lower development time and reduce costs. The LwM2M specification provides great flexibility, but requires more forethought in the design and customization of the client compared to MQTT.

[2] Initial connection refers to the initial registration of an IoT device to an IoT platform, as well as the connection made after a device reboot.

[3] It can be argued that the 4% efficiency of MQTT over LwM2M is insignificant. MachNation supports the results of this whitepaper based on the technology selected for testing as shown on Figure 3. In addition, it is worth mentioning that an interrupted OTA download over the MQTT client used in this study would require re-transmission of an entire firmware image, whereas the out-of-the-box LwM2M client used in this study supports resumption of the download from point of interruption. This could have significant implications on download time and amount of data transferred in favor of LwM2M. MachNation recommends additional testing of OTA updates using different clients and devices.

[4] MachNation did not test longer intervals (e.g., 1 update per day) for the power consumption test. It is possible that test results of longer intervals would show similar findings to the tests of shorter intervals.

[5] While there are only a few reference implementations of LwM2M v1.0 on GitHub, there are many chipset vendors and module manufacturers that support LwM2M for their own products. See footnote 18 for more information.

Introduction

Enterprises continue to deploy IoT solutions for a variety of solutions. Some of these solutions, like connected robotics, autonomous driving, drone management, and more, produce tremendous amounts of data, rely on streaming analytics services, and require solutions that both monitor and control the IoT asset. Other solutions, like smart parking, small asset tracking, cold-chain management, and others, produce small quantities of data, rely on analytics from a data historian, and require solutions that primarily accomplish condition monitoring of IoT assets.

MachNation has spoken with many enterprises that are debating the use of LwM2M and MQTT for their IoT deployments. MachNation discovered three common themes. First, these enterprises do not know which protocol will yield them more device efficiency and performance benefits. Second, enterprises confirmed the lack of rigorous comparison between the two protocols to help them select the right one for their IoT applications. Third, enterprises recognize that the specific requirements of an IoT solution should dictate the selection of device management and communications protocol.

Lightweight M2M (LwM2M) is a protocol from the Open Mobile Alliance[6] for IoT device management. It is designed for remote management of M2M devices and related service enablement, including real time telemetry and command and control; features a modern architectural design based on REST; defines an extensible resource and data model often referred to as the Smart Objects model; and builds on a secure data transfer standard called the Constrained Application Protocol (CoAP).[7] LwM2M comes with a set of standardized management objects as well as many others that can be added as standard objects based on the IPSO Alliance's IPSO Smart Object Registry.[8] Additionally, LwM2M supports a variety of data encoding formats for device-to-platform and platform-to-device communication, including binary type-length-value (TLV), JavaScript Object Notation (JSON), and Concise Binary Object Representation (CBOR).

MQTT is a machine-to-machine (M2M) and IoT connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport to aid in efficient, standardized message delivery.[9] Although the MQTT protocol itself defines no specific standard for data encoding, most implementations leverage JSON for device-to-platform and platform-to-device communication. While this is not a technical limitation of the protocol itself and other encodings such as CBOR or TLV are possible, they are used much less regularly in practice largely due to lack of platform support. Additionally, platforms such as AWS IoT and Azure IoT require telemetry messages to be in a JSON format to support many typical out-of-the-box functionalities such as digital twins or pre-integrated event processing. The use of non-JSON encoding schemas typically requires additional data-translation solutions to be implemented before the inbound messages can be stored or processed.

Today, MQTT is the *de facto* communications protocol used by enterprises deploying IoT solutions. While these enterprises might investigate other protocols during creation of a proof of concept (POC), often they will select MQTT due to the relatively strong ecosystem support that has existed for many years around MQTT. For example, hyper-scale cloud vendors like Amazon, Google, and Microsoft, support and market MQTT, offer productized MQTT clients, and supply documentation to support an enterprise's MQTT implementation. These vendors do not support LwM2M as of 1Q2020.

Much confusion exists—driven by lack of replicable, quantitative data—about the relative efficiencies and performance of LwM2M and MQTT. So with technology support from AT&T and AVSystem, MachNation undertook an in-lab experiment to quantify and compare the benefits of LwM2M and MQTT clients on a typical IoT device.

[6] OMA is an organization supported by AT&T, other wireless carriers, device manufacturers, and IoT platform vendors. These parties work together to ensure solid and efficient clients are built and certified for the industry.
[7] For more information, please see https://www.omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/. Also please note that Datagram Transport Layer Security (DTLS) wrapping of CoAP creates the security element of the protocol.
[8] Details can be found at the OMA LwM2M Object and Resource Registry at http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html
[9] For more information, please see http://mqtt.org/

Methodology and Technical Specifications

MachNation began this hands-on benchmarking study of LwM2M and MQTT in 2Q2019. The study was completed in 4Q2019 after 7 months of test design, implementation, and testing. The testing was performed at MachNation's USA test lab in metropolitan Boston.

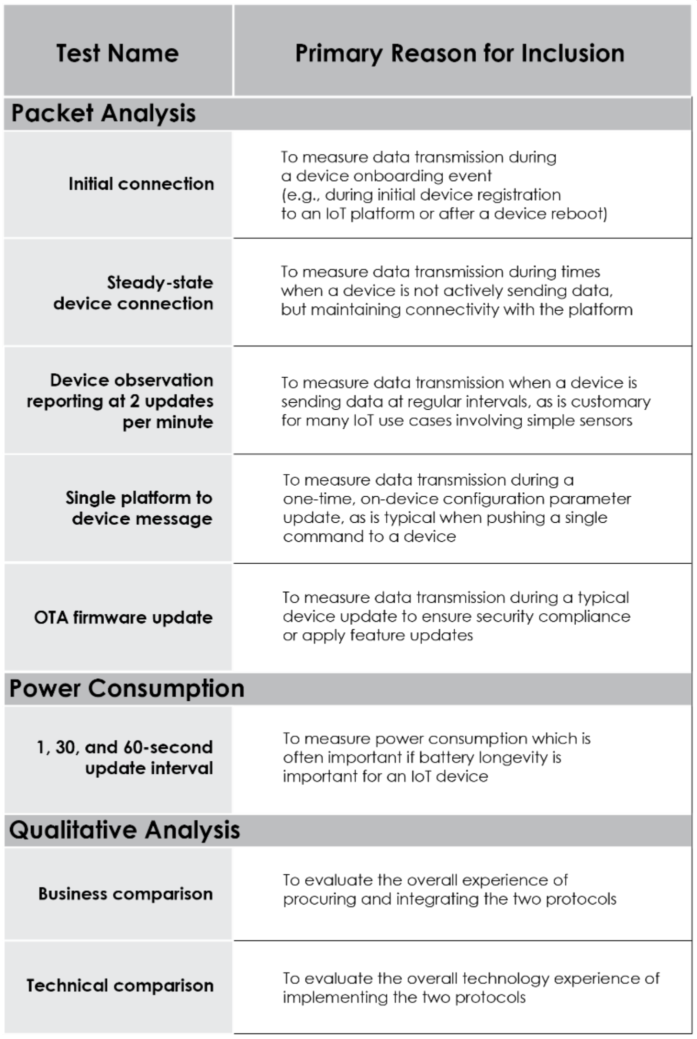
MachNation completed the following steps for testing:

1. Installed a LwM2M Anjay-based client and Amazon AWS IoT MQTT client on a Raspberry Pi 4 Model B
2. Designed and conducted a set of efficiency and performance tests under multiple test scenarios
3. Ingested, normalized, and aggregated raw test data
4. Drafted the results and implications in this whitepaper

MachNation supplied testing services including device set-up, platform set-up, client implementation, metric collection, and analysis; was responsible for drafting this whitepaper; and supplied test equipment as needed for this analysis. AT&T and AVSystem, as study sponsors, supplied technology and technology advice to support testing. Both AT&T and AVSystem reviewed drafts of this whitepaper and provided edits and guidance.

MachNation chose tests that would provide meaningful, real-world insights for enterprises making the selection of IoT protocols. While there are many tests that could be helpful, MachNation chose a subset of tests that represent typical activities of IoT devices and platforms like initial connection between an IoT device and platform, pushing a firmware update to a device, and streaming data every 30 seconds from a device to a platform. See Figure 2 summarizing the chosen tests and primary reason for inclusion.

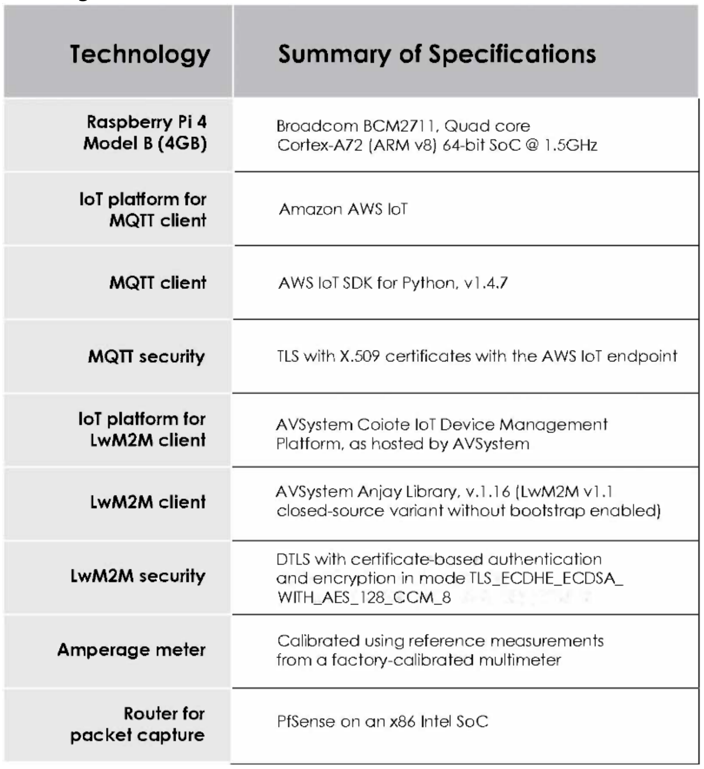
FIGURE 2: IOT PROTOCOL TESTS AND PRIMARY REASON FOR INCLUSION [SOURCE: MACHNATION, 2020]



Test Name	Primary Reason for Inclusion
Packet Analysis	
Initial connection	To measure data transmission during a device onboarding event (e.g., during initial device registration to an IoT platform or after a device reboot)
Steady-state device connection	To measure data transmission during times when a device is not actively sending data, but maintaining connectivity with the platform
Device observation reporting at 2 updates per minute	To measure data transmission when a device is sending data at regular intervals, as is customary for many IoT use cases involving simple sensors
Single platform to device message	To measure data transmission during a one-time, on-device configuration parameter update, as is typical when pushing a single command to a device
OTA firmware update	To measure data transmission during a typical device update to ensure security compliance or apply feature updates
Power Consumption	
1, 30, and 60-second update interval	To measure power consumption which is often important if battery longevity is important for an IoT device
Qualitative Analysis	
Business comparison	To evaluate the overall experience of procuring and integrating the two protocols
Technical comparison	To evaluate the overall technology experience of implementing the two protocols

MachNation chose a variety of technology to conduct testing of LwM2M and MQTT clients. MachNation sought to choose industry-available technology that would be accessible to a typical enterprise developer seeking to create an IoT solution for a POC. See Figure 3 summarizing the technology used for testing.

FIGURE 3: TECHNOLOGY USED FOR LWM2M AND MQTT TESTING [SOURCE: MACHNATION, 2020]



Technology	Summary of Specifications
Raspberry Pi 4 Model B (4GB)	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
IoT platform for MQTT client	Amazon AWS IoT
MQTT client	AWS IoT SDK for Python, v1.4.7
MQTT security	TLS with X.509 certificates with the AWS IoT endpoint
IoT platform for LwM2M client	AVSystem Coiote IoT Device Management Platform, as hosted by AVSystem
LwM2M client	AVSystem Anjay Library, v.1.16 (LwM2M v1.1 closed-source variant without bootstrap enabled)
LwM2M security	DTLS with certificate-based authentication and encryption in mode TLS_ECDHE_ECDSA_ WITH_AES_128_CCM_8
Amperage meter	Calibrated using reference measurements from a factory-calibrated multimeter
Router for packet capture	PfSense on an x86 Intel SoC

MachNation and the sponsors sought to create a baseline comparison to provide a fair, quantitative, and test-based assessment of the two protocols. The goal was to design a comparison that would emulate, as much as possible, a true-to-life deployment of an IoT solution using LwM2M and MQTT that would neither benefit nor penalize either protocol based on the test design methodology. Below we discuss some additional test design choices.

Device Clients

MachNation chose comparable IoT device clients for testing. MachNation chose[10] the Amazon AWS IoT SDK for Python v1.4.7 for the MQTT tests. MachNation programmed the MQTT client to use AWS IoT's native "jobs" service for managing firmware over-the-air operations, as well as monitoring a specific MQTT topic to configure the reporting frequency of the northbound data observations (i.e., northbound messages). Observation updates[11] were performed using the native AWS IoT device shadow functionality[12], as this is the reference implementation

[10] Choice of IoT technology and vendors' particular implementation of that technology can have meaningful impacts on test results. In this study, MachNation only tested vendors' technology as shown on Figure 3. MachNation did not test technology stacks, such as LTE-M or NB-IoT, which may or may not provide different test results. In addition, MachNation chose an Anjay-based LwM2M and an Amazon AWS MQTT client as shown on Figure 3. A discussion of additional areas of testing exploration is covered in the whitepaper section entitled, "Areas of future suggested research".

[11] MQTT does not use the concept of observe or notify, only publish or subscribe. However, AWS' IoT implementation, when leveraging its Device Shadow Service, does use the observe or notify concept. It is roughly analogous to observation reporting for LwM2M.

[12] The concept of a device shadow is roughly analogous to the concept of a device twin or digital twin. Regardless of the terminology, using a device shadow or digital twin is the most common approach to send telemetry into a managed pipeline within the northbound platform. Some customers may choose to simply leverage the message broker directly, manually consume broker-ingested messages, and store/process them as needed. However, for most IoT platform out-of-the-box

for managing device-to-platform observation messages and will likely be the first point of implementation for many typical customer use cases. JSON was leveraged as the encoding schema for messaging.

MachNation chose AVSystem Anjay Library v.1.16 closed-source variant for the LwM2M scenarios without bootstrap enabled. This client uses version 1.1 of the LwM2M protocol. MachNation, with assistance from AVSystem, programmed the LwM2M client to use the default "firmware update" object (/5/) to execute firmware update operations and used native LwM2M features (pmax/pmin) to configure the reporting frequency of the northbound data observations. Binary TLV was leveraged as the encoding schema for messaging.

Security

To ensure a typically designed IoT implementation, MachNation chose TLS with X.509 certificates for MQTT connectivity with the AWS IoT endpoint. DTLS with certificate-based authentication was leveraged for LwM2M connectivity over CoAP/UDP with the AVSystem Coiote IoT Device Management platform, with the exception of one firmware OTA test that leveraged HTTPS for binary delivery from the platform to the device.

Packet Capture

During the testing process packets were captured using tcpdump on a pfSense router upstream of the test device. Packets were filtered to only count UDP or TCP packets carrying traffic to and from the device and platform. DNS lookups, ARP requests, and other network traffic were not included in the packet or byte counts. The MQTT client used MQTT/TCP for observations and commands while leveraging HTTPS for firmware file retrieval. The LwM2M client communicated only over CoAP/UDP throughout the tests.

Data Observations

MachNation configured both IoT device clients to send a four-parameter message that included a string, an integer, a float, and a Boolean observation as a single object and message.

Invariably, in any lab-based comparison, it is impossible to create a completely apples-to-apples comparison. There are always decisions that testers need to make that might influence the outcome of the analysis. MachNation will continue to review its comparison and modify assumptions to better reflect real-world IoT deployment use cases.

functionality, such as device state monitoring, accessing the on-platform rules engines, on-platform data transformation, leveraging the automatic data historian, and others, a developer must leverage a platform's device shadow or digital twin model. Additionally, the device shadow is the reference implementation method for AWS IoT, as provided in its developer documentation.

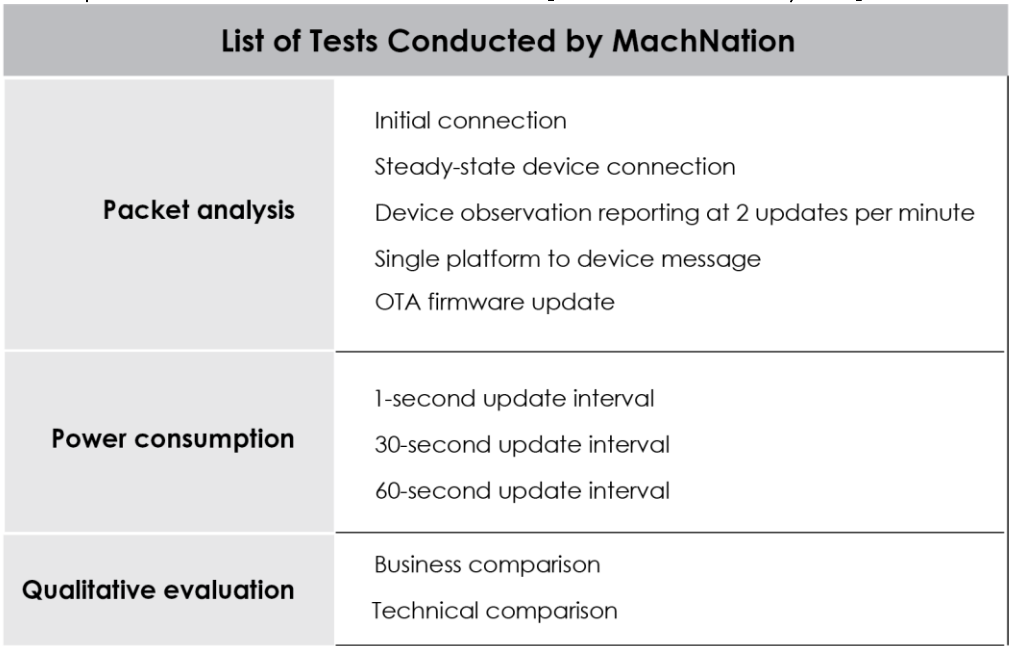
Test Findings

MachNation completed a set of hands-on tests comparing the efficiency and performance metrics on a typical IoT device using LwM2M and MQTT. In this section, we will

- Define each test
- Describe why we included the test
- Present the efficiency and performance data
- Summarize the implications of the findings.

See Figure 4 for the list of tests conducted by MachNation as part of this study.

FIGURE 4: LIST OF TESTS CONDUCTED BY MACHNATION [SOURCE: MACHNATION, 2020]



List of Tests Conducted by MachNation	
Packet analysis	Initial connection Steady-state device connection Device observation reporting at 2 updates per minute Single platform to device message OTA firmware update
Power consumption	1-second update interval 30-second update interval 60-second update interval
Qualitative evaluation	Business comparison Technical comparison

Packet analysis: initial connection[13]

LwM2M is 72% more efficient than MQTT at delivering data during the initial connection between IoT device and platform.

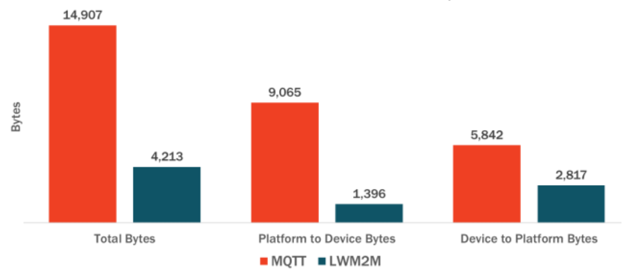
In this scenario, MachNation tested the delivery of packets during the initial connection between the IoT device and platform. For MQTT, we tested the initial connection from the device to the AWS IoT MQTT broker, including a single shadow update upon connection. For LwM2M, we tested an initial non-bootstrap connection of the device to the AVSystem Coiote LwM2M endpoint, including a variety of LwM2M messages, such as optimized on-device object discovery and an initial reporting of the custom observation object. Overall, the packet capture window was 2 minutes.

MachNation chose to include a packet analysis for the initial connection to highlight the impact on an IoT network and device and to illustrate the differences in typical implementations for a device onboarding event (i.e., device registration to an IoT platform or device reboot) in MQTT- and LwM2M-based solutions.

Overall, MachNation found that LwM2M is 72% more efficient at delivering data during the initial connection between IoT device and platform (or after device reboot) than MQTT. During the initial connection, LwM2M and MQTT required 4213 bytes and 14907 bytes, respectively, with the average LwM2M packet size being 9% smaller than the average MQTT packet size.

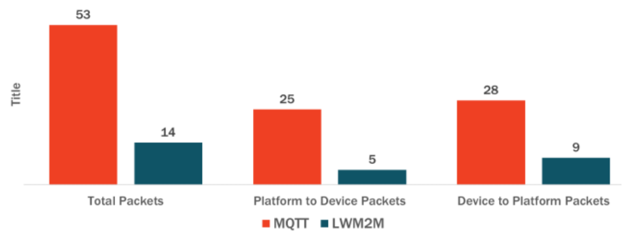
Below we present the data from the initial connection scenario.

FIGURE 5-1: TOTAL BYTES TRANSFERRED DURING INITIAL DEVICE CONNECTION [SOURCE: MACHNATION, 2020]



As shown on Figure 5-1, during the initial connection between IoT device and platform, LwM2M required 72% fewer bytes of data than MQTT. Approximately 60% of the MQTT data and 33% of the LwM2M data traveled from platform to device, illustrating the relative inefficiency of JSON messages over MQTT compared to binary TLV messages over LwM2M during registration to an IoT platform. The remaining 40% and 67% for MQTT and LwM2M, respectively, were data traffic from the device to the platform.

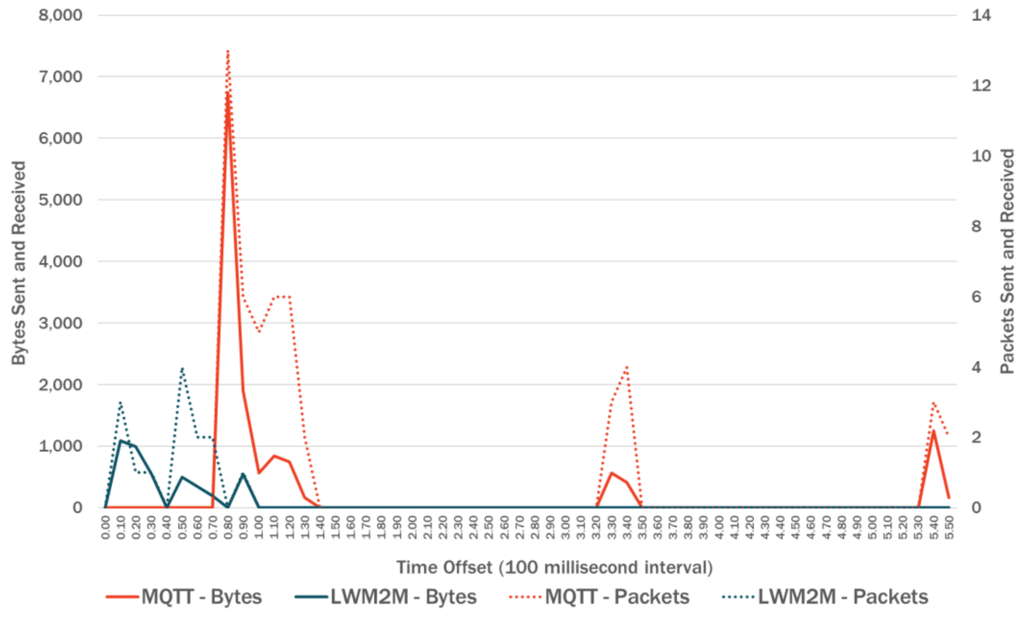
FIGURE 5-2: TOTAL PACKETS TRANSFERRED DURING INITIAL DEVICE CONNECTION [SOURCE: MACHNATION, 2020]



As shown on Figure 5-2, during the initial connection between IoT device and platform, LwM2M delivered 74% fewer packets of data than MQTT, supporting the overall efficiency of LwM2M. Of additional note, the LwM2M client transmitted more parameters to the platform during the initial connection process compared to the MQTT client, but nonetheless achieved this with fewer packets and fewer bytes.

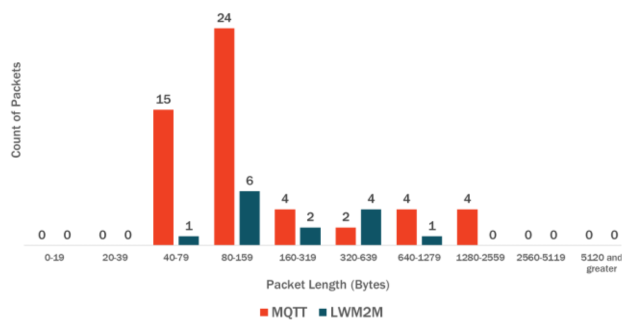
[13] Initial connection refers to the initial registration of an IoT device to an IoT platform, as well as the connection made after a device reboot.

FIGURE 5-3: BYTES AND PACKETS TRANSFERRED DURING INITIAL DEVICE CONNECTION OVER TIME [SOURCE: MACHNATION, 2020]



The fact that fewer packets were sent during the initial connection phase is supplemented by data from Figure 5-3 showing that MQTT data tends to be more bursty (i.e., transmitting a higher total throughput of messages in a given time window) during initial connection than LwM2M. This implies that network planning during the initial connection phase is important in cases where large numbers of MQTT devices are brought on-line simultaneously. LwM2M does not exhibit this type of bursty packet behavior, likely leading to less complex scaling challenges during large-scale onboarding of IoT devices. The large MQTT peaks are likely caused by the AWS Device Shadow Service updating the platform with data from the device. It is possible that other MQTT clients would not exhibit the same type of peaks, although MachNation did not test non-AWS MQTT clients as part of this study.

FIGURE 5-4: HISTOGRAM OF PACKET LENGTHS DURING INITIAL DEVICE CONNECTION [SOURCE: MACHNATION, 2020]



As shown on Figure 5-4, during the initial connection between IoT device and platform, LwM2M packet lengths tended to be less variable than MQTT packet lengths for the two chosen clients for testing.[14] The average LwM2M package length was 295 bytes, 9% less than the average MQTT packet length of 324 bytes. In particular, LwM2M may be the preferential choice for networks with constrained message transmission unit (MTU) sizes.

[14] Theoretically, MQTT message size can be constrained through changes during device implementation, however, MachNation believes this would be a bit non-standard. We believe a developer would not make these changes unless there is a very specific need. MachNation conducted testing as these two clients would be used in real-world environments, therefore, we believe the conclusions drawn in this section are reasonable.

Packet analysis: steady state without observation reporting

LwM2M devices transmit 31% less data in a steady state than MQTT devices.

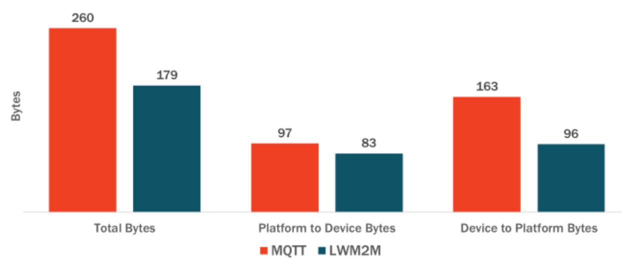
In this scenario, MachNation tested the packet delivery during a steady-state connection without observation reporting between an IoT device and platform. For MQTT, we tested the AWS IoT SDK device client connected to the AWS IoT broker with no explicit messages or observations sent across the connection. For LwM2M, we tested the Anjay client connected to the AVSystem Coiote IoT Device Management platform with no observations or reporting intervals set in the LwM2M client or platform. Overall, the packet capture window was 10 minutes.

MachNation chose to include a packet analysis for the steady state, because many IoT devices, especially those collecting data based on a specific triggering parameter or rule, remain in a state where the device is mostly listening for events or monitoring behavior without actively reporting any observations. Understanding the transmittal of data and packets (sometimes called chattiness) during these times helps customers estimate ongoing networking costs and allows network planners to design appropriate IoT networks.

Overall, MachNation found that a LwM2M delivered 179 bytes in a total of 2 packets and MQTT delivered 260 bytes in a total of 3 packets. The MQTT heartbeats were a result of the AWS IoT SDK leveraging the device shadow functionality, as would be typical for many customers implementing AWS IoT, although this is not easily configured.[15] In contrast, the LwM2M client has an operator-configurable "heartbeat" period, which was set to 10-minutes, resulting in a single heartbeat capture during our test.

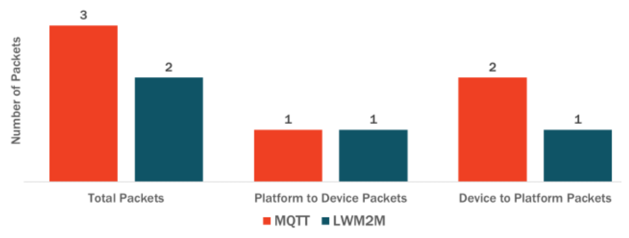
Below we present the data from the steady-state scenario.

FIGURE 6-1: TOTAL BYTES TRANSFERRED DURING DEVICE STEADY-STATE [SOURCE: MACHNATION, 2020]



As shown on Figure 6-1, the LwM2M client delivered 179 bytes of data, whereas the MQTT client delivered 260 bytes during the steady-state scenario. Approximately 63% of the MQTT and 54% of the LwM2M data traveled from device to platform, showing that the majority of steady-state communications was initiated by the device verifying communication with the platform.

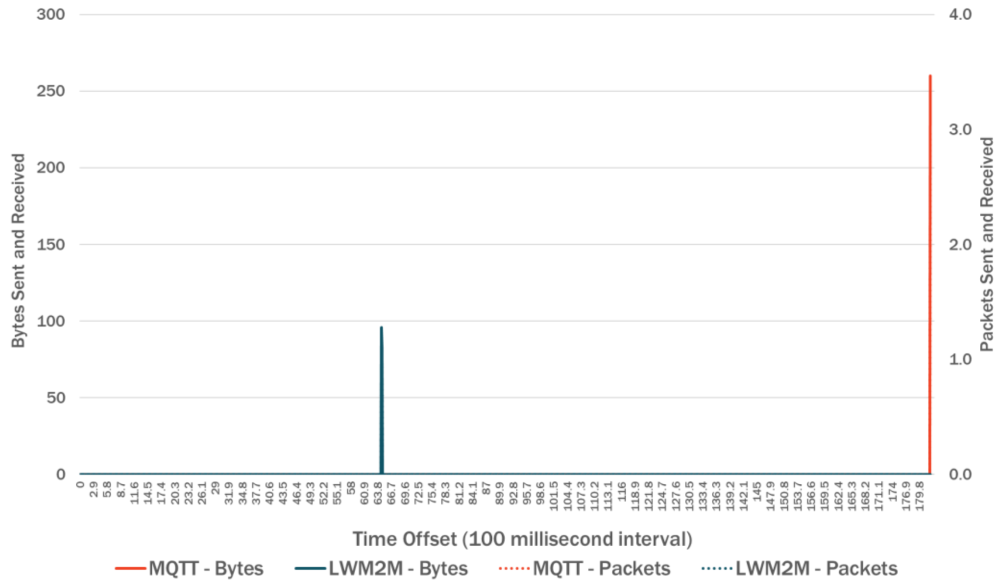
FIGURE 6-2: TOTAL PACKETS TRANSFERRED DURING DEVICE STEADY-STATE [SOURCE: MACHNATION, 2020]



As shown on Figure 6-2, during the steady-state of device-to-platform communication, LwM2M delivered 2 packets of data, whereas MQTT delivered 3.

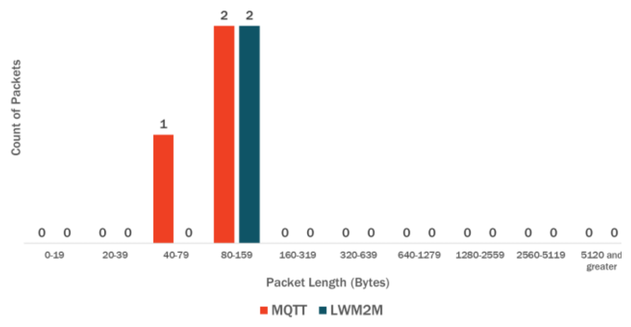
[15] Heartbeats are typical to nearly all IP-connected IoT devices, but may vary by vendor implementation. MachNation did not test other vendors' IoT implementations, only the listed vendors in this particular whitepaper. MachNation did not test technology stacks, such as LTE-M or NB-IoT, which may or may not provide different test results. Discussion of follow-on testing is covered in the whitepaper section entitled, "Areas of future suggested research".

FIGURE 6-3: BYTES AND PACKETS TRANSFERRED DURING DEVICE STEADY-STATE OVER TIME [SOURCE: MACHNATION, 2020]



As shown on Figure 6-3, interestingly the MQTT device delivered all data in the last 100 milliseconds of the capture window, indicating the heartbeat interval appears to be around 10 minutes. As we were able to manually define the LwM2M "heartbeat" interval, we were able to offset the timing of the test to capture the message roughly during the middle of the test window.

FIGURE 6-4: HISTOGRAM OF PACKET LENGTHS DURING DEVICE STEADY-STATE [SOURCE: MACHNATION, 2020]



As shown on Figure 6-4, during the steady-state scenario, the average MQTT and LwM2M package lengths were 100 bytes and 120 bytes, respectively. While the average LwM2M packet length was greater than the MQTT packet length, the number of total packets was less for LwM2M, as has already been described.

Packet Analysis: observation reporting at 30-second reporting intervals

LwM2M is 88% more efficient than MQTT at delivering data during observation reporting at 30-second intervals.

In this scenario, MachNation tested the delivery of packets with observation reporting at 30-second intervals. MachNation chose a 30-second interval as indicative of a realistic, real-world solution. The actual interval length (30-seconds) is not overly relevant: MachNation believes that other interval lengths would yield the same results in terms of the percentage efficiency of the LwM2M client over the MQTT client. For purposes of this whitepaper, MachNation only tested a 30-second interval. Furthermore, most observation reporting windows are time-bounded in one way or another. Very few real-world IoT implementations have no heartbeat (minimum reporting interval) and rely solely on device-initiated observations. Therefore, MachNation chose to include this scenario as representative of a real-world IoT use case.

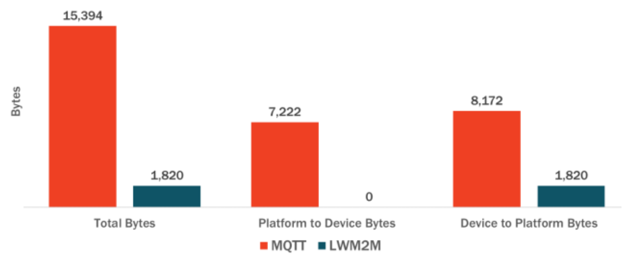
For MQTT, we tested the AWS client on the AWS IoT platform configured to send a device shadow update of one string, one floating-point number, one integer, and one Boolean every 30 seconds (i.e., 2 observations per minute). For LwM2M, we tested the Anjay client connected to the Coiote IoT Device Management platform with a reporting interval of 30 seconds for a single custom LwM2M object containing the same set of data as the MQTT observations. Overall, the packet capture window was 5 minutes, yielding 10 observations each captured for LwM2M and MQTT.

MachNation chose to include this packet analysis to simulate a device sending data at regular intervals as is customary for many IoT use cases involving simple sensors.

Overall, MachNation found that LwM2M is 88% more efficient than MQTT at delivering data during observation reporting at 30-second intervals. During the test, LwM2M and MQTT delivered 15394 and 1820 bytes of data, respectively, with the average LwM2M packet size being 64% smaller than the average MQTT packet size.

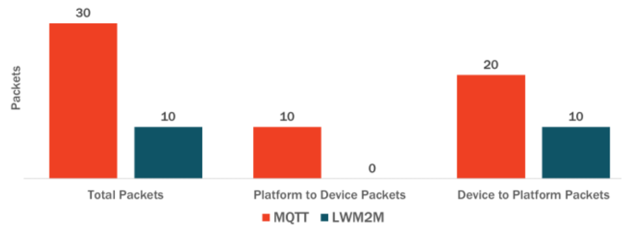
Below we present the data from the 30-second interval observation reporting scenario.

FIGURE 7-1: TOTAL BYTES TRANSFERRED DURING TESTS FOR 30-SECOND REPORTING INTERVALS [SOURCE: MACHNATION, 2020]



As shown on Figure 7-1, during observation reporting at 30-second intervals, LwM2M delivered 88% fewer bytes of data than MQTT. 47% of the MQTT data and 0% of the LwM2M data, traveled from platform to device, while 53% of the MQTT data and 100% of the LwM2M data traveled from IoT device to the platform.

FIGURE 7-2: TOTAL PACKETS TRANSFERRED DURING TESTS FOR 30-SECOND REPORTING INTERVALS
[SOURCE: MACHNATION, 2020]



As shown on Figure 7-2, during observation reporting at 30-second intervals, LwM2M delivered 67% fewer packets of data than MQTT. Furthermore, all of the LwM2M packets were sent from IoT device to platform, whereas 67% of the MQTT packets were sent from device to platform. In particular, we note the additional overhead of the MQTT protocol when used with AWS IoT's Device Shadow Service, requiring an acknowledgement message for each observation sent to the platform from the device. Different results are possible had MachNation chosen other reference implementations.

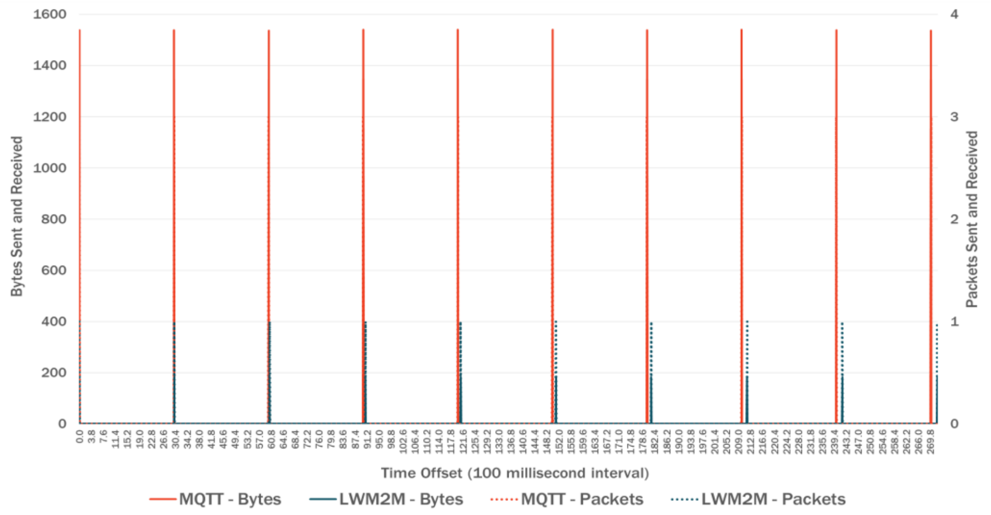
FIGURE 7-3: MQTT AND LWM2M PACKETS AND LENGTHS DURING A SINGLE OBSERVATION REPORT
[SOURCE: MACHNATION, 2020]

Action	Number of Packets	Packet Length (bytes)
MQTT		
Device sends the shadow update to the platform	1	Approximately 750
Platform confirms the shadow update	1	Approximately 720
Device confirms update of the shadow on the platform	1	Approximately 66
LwM2M		
Device sends update to the platform	1	182

In Figure 7-3, we present a more detailed look into the total number of packets and bytes, respectively, sent and received by both the MQTT and LwM2M clients during a single observation report to the platform from the device. As previously described, we can clearly see the additional overhead of the AWS IoT device shadow service, where the device sends the shadow update to the platform delivering approximately 750 bytes, the platform confirms the shadow update delivering approximately 720 bytes, and the device confirms update of the shadow on the platform delivering approximately 66 bytes. As previously mentioned, it is possible that non-AWS MQTT clients would exhibit different amounts of data delivery, although MachNation did not test other MQTT clients as part of this study.

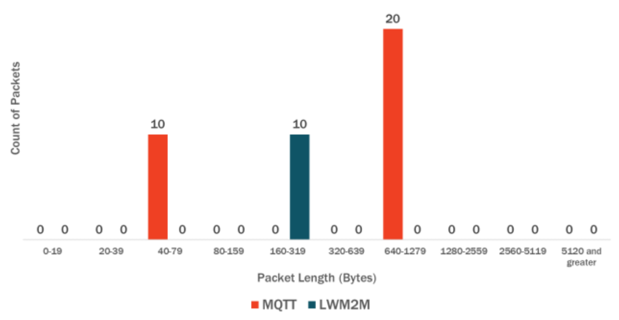
In the LwM2M-equipped device only a single packet of 182 bytes is sent with a consistent packet length throughout the test.

FIGURE 7-4: BYTES AND PACKETS TRANSFERRED WITH A 30-SECOND REPORTING INTERVAL OVER TIME [SOURCE: MACHNATION, 2020]



As shown on Figure 7-4 and supporting the detailed analysis on Figure 7-3, both LwM2M and MQTT deliver a consistent number and size of packets every 30 seconds. Each observation yields 3 MQTT packets of 1536 bytes and 1 LwM2M packet of 182 bytes. Overall, LwM2M is 88% more efficient in terms of packet size and 67% more efficient in terms of the number of packets sent per observation, irrespective of the reporting interval chosen.

FIGURE 7-5: HISTOGRAM OF PACKET LENGTHS DURING TESTS FOR 30-SECOND REPORTING INTERVALS [SOURCE: MACHNATION, 2020]



As shown on Figure 7-5, during the 30-second observation reporting scenario, LwM2M packet lengths were highly consistent whereas MQTT packet lengths varied[16] based on whether data were going from device to platform or vice versa. Overall, the average packet length for LwM2M was 88% less than for MQTT during the 30-second observation reporting scenario.

[16] The variance in packet lengths is specific to the vendor's (AWS') implementation of MQTT. Other vendors' implementations might yield different results.

Packet analysis: single platform-to-device message

LwM2M is 17% more efficient than MQTT at delivering data during a single platform-to-device message.

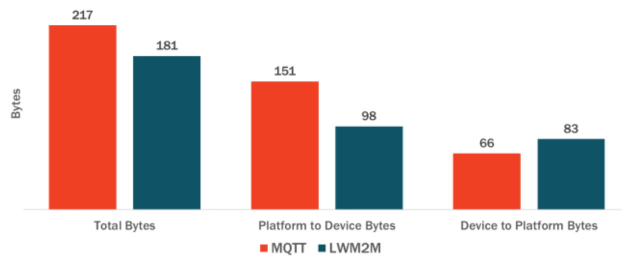
In this scenario, MachNation tested a single update of an on-device variable initiated from the IoT platform. For MQTT, we tested a platform-initiated update of a single integer variable (i.e., a counter) via an MQTT message passed directly from the broker to the device client. For this test, MachNation did not leverage AWS IoT's device shadow model[17], but instead configured the device client to listen to a specific topic. For LwM2M, MachNation tested a server-initiated update of the integer (i.e., a counter) within the custom LwM2M object. Overall, the packet capture window was 2 minutes.

MachNation chose to include a packet analysis for a single platform-to-device message to simulate a one-time, on-device parameter update, because it is a typical task to adjust an on-device configuration parameter or push a single command to a connected device.

Overall, MachNation found that LwM2M is 17% more efficient than MQTT at delivering data during a single platform-to-device message. During the test, MQTT and LwM2M delivered 217 and 181 bytes of data, respectively. This test is useful in comparing protocol efficiency and overhead, because each protocol transmits an update of the smallest possible size. One might expect LwM2M to provide better efficiency for longer and more complex observations or other messages where the MQTT/TCP overhead is more severe, but in real-world scenarios, most IoT devices should be optimized to only transmit the smallest amount of data required to accomplish IoT business objectives. LwM2M offers several advantages in this area, providing more flexible and granular control over the exact type and content of messages exchanged between the device and the platform.

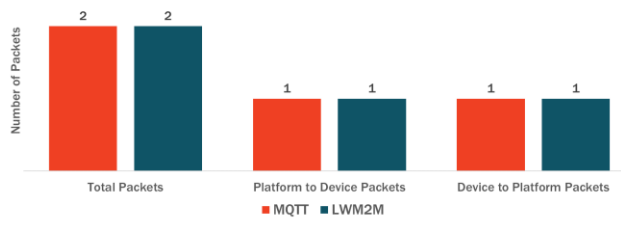
Below we present the data from the single platform-to-device message scenario.

FIGURE 8-1: TOTAL BYTES TRANSFERRED DURING TESTS FOR A SINGLE PLATFORM-TO-DEVICE MESSAGE [SOURCE: MACHNATION, 2020]



As shown on Figure 8-1, during a single platform-to-device message push, LwM2M delivered 17% fewer bytes of data than MQTT, although both solutions were quite efficient in this test. Approximately 70% of the MQTT and 54% of the LwM2M data traveled from platform to device, showing that the majority of data during a single message session was initiated by the platform requesting some action on the part of the device.

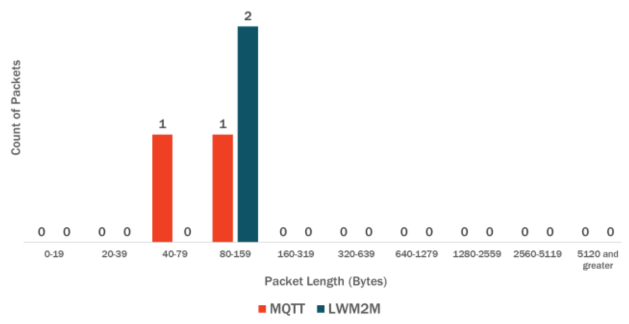
FIGURE 8-2: TOTAL PACKETS TRANSFERRED DURING TESTS FOR SINGLE PLATFORM-TO-DEVICE MESSAGE [SOURCE: MACHNATION, 2020]



As shown in Figure 8-2, both the MQTT and LwM2M client required 2 messages, namely 1 from the platform initiating the update and 1 from the device confirming the change. Although typical AWS IoT MQTT implementations may leverage the device shadow service with its additional messaging overhead for the management and reporting of data observations, MachNation's test is more indicative of a simple configuration update or platform-to-device command.

[17] According to AWS' documentation, the device shadow model is not suggested for usage as a command-and-control messaging service.

FIGURE 8-3: HISTOGRAM OF PACKET LENGTHS DURING TESTS FOR SINGLE PLATFORM-TO-DEVICE MESSAGE [SOURCE: MACHNATION, 2020]



As shown on Figure 8-3, during the single platform-to-device messaging test, LwM2M packet lengths were of consistent size whereas MQTT packet lengths varied based on whether data were going to or from the device. Overall, the average packet length for LwM2M was 17% less than for MQTT during the single platform-to-device messaging scenario.

Packet analysis: OTA firmware update

MQTT is 4% more efficient than LwM2M at delivering data during an over-the-air (OTA) firmware update.

In this scenario, MachNation tested a single OTA firmware update of an IoT device. For MQTT, we tested a platform-initiated OTA update to the device firmware leveraging the AWS IoT "jobs" functionality, as would be typical for a real-world IoT implementation. For the MQTT client, the update file was stored in an AWS S3 file storage service, retrieved or pulled by the device over the HTTPS (i.e., HTTP and TLS) protocol, and communicated its job status via MQTT.

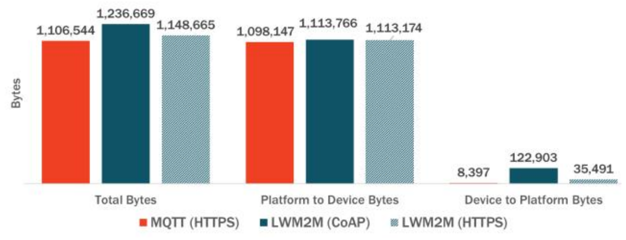
For LwM2M, MachNation tested two implementations of a similar platform-initiated OTA firmware update. In the first test, we leveraged the standard LwM2M firmware object (/5/) with the file delivered from the Coiote platform to the device over CoAP with DTLS via the pull method. In the second LwM2M test, we leveraged HTTPS (using TLS over HTTP) via the "pull" method for delivery of the firmware update itself, while maintaining the CoAP channel for all other communication. The firmware update for both MQTT and LwM2M clients was an identical randomly-filled 1,048,576 byte test file. Though the firmware update was not actually applied to the devices, both clients were configured to report a successful completion of the update task after retrieving the file. Overall, the packet capture window was 5 minutes.

MachNation chose to include a firmware OTA test to simulate the typical type of update completed on IoT devices to ensure ongoing security compliance or feature updates and improvements.

Overall, MachNation found that MQTT is 4% more efficient than LwM2M over HTTPS at delivering data during a firmware OTA update. During the scenario, LwM2M and MQTT used 1.15 and 1.11 megabytes of data, respectively.

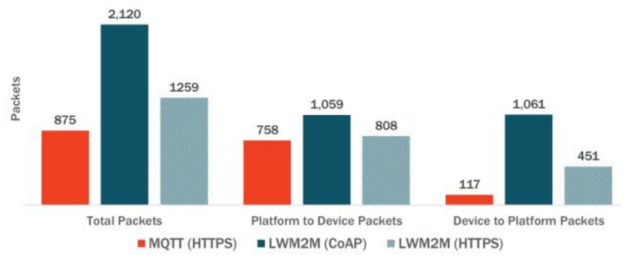
Below we present the data from the OTA firmware update test.

FIGURE 9-1: TOTAL BYTES TRANSFERRED DURING OVER-THE-AIR FIRMWARE UPDATE
[SOURCE: MACHNATION, 2020]



As shown on Figure 9-1, during an OTA firmware update, the MQTT client leveraging HTTPS for file delivery consumed 4% less data than the LwM2M client over HTTPS and 11% less data than the LwM2M client over CoAP. In all test cases, almost all of the data traveled from platform to device, as expected for firmware updates. And as expected, the data transferred was close to 1.05 megabytes (MB), the size of the test file delivered to simulate a firmware update.

FIGURE 9-2: TOTAL PACKETS TRANSFERRED DURING OVER-THE-AIR FIRMWARE UPDATE
[SOURCE: MACHNATION, 2020]



As shown on Figure 9-2, during the OTA firmware update, MQTT delivered 30% fewer packets of data than LwM2M over HTTPS. The disparity between total packets transferred is most likely due to the increased number of messages exchanged between the LwM2M client and the Coiote platform before and after the update process. However, despite the increased total number of messages, as seen in the previous figure, Figure 9-1, the overall impact on total bytes transferred is nearly negligible.

Figure 9-3a: Bytes and packets transferred during LwM2M (CoAP) over-the-air firmware update time [Source: MachNation, 2020]

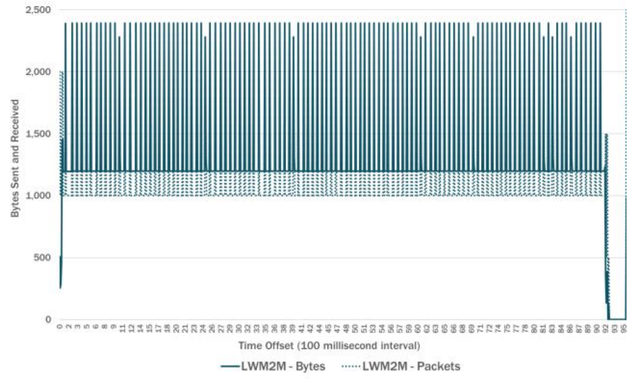


Figure 9-3b: Bytes and packets transferred during MQTT over-the-air firmware update time [Source: MachNation, 2020]

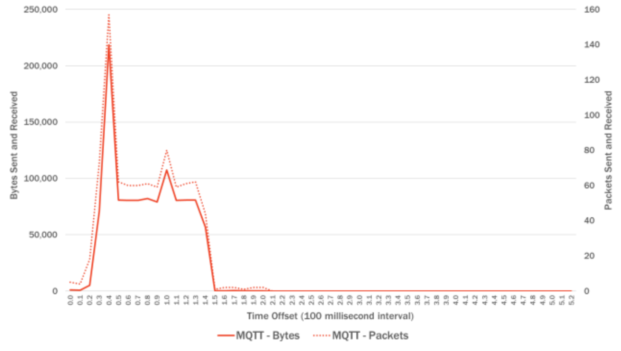
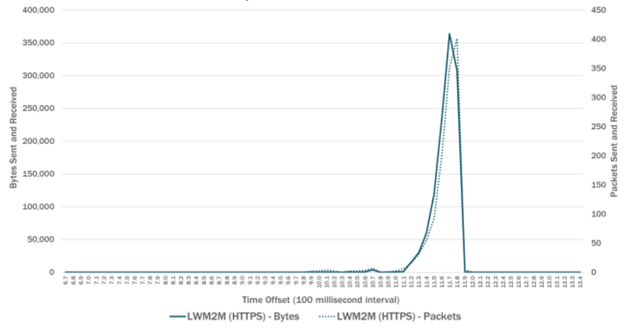
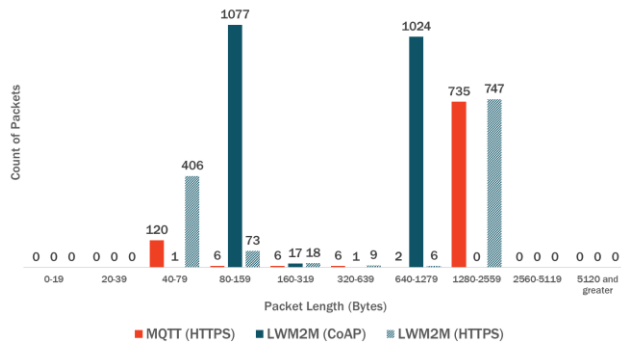


Figure 9-3c: Bytes and packets transferred during LwM2M (HTTPS) over-the-air firmware update time (truncated time frame) [Source: MachNation, 2020]



As shown on Figures 9-3a, 9-3b, and 9-3c, the data and packet transfer patterns over time vary across the two clients. The LwM2M client leveraging CoAP (Figure 9-3a) sent small individual packets over a greater length of time compared to the other tests. Leveraging HTTPS for the transfer of the firmware file (Figure 9-3b), the MQTT client quickly retrieved the entire firmware file from AWS S3, then reported its successful update status to the AWS IoT jobs service via MQTT. The LwM2M client leveraging HTTPS (Figure 9-3c) exhibited a transfer similar to the MQTT over HTTPS client, although with a more pronounced spike. It is worth mentioning that AWS' MQTT client does not provide the same resilience to interrupted communication during a FOTA process as the LwM2M client. Therefore, an interrupted FOTA download over AWS MQTT would require re-transmission of the entire firmware image, whereas the out-of-the-box LwM2M client supports resumption of the download from point of interruption. This could have significant implications on download time and amount of data transferred.

FIGURE 9-4: HISTOGRAM OF PACKET LENGTHS DURING OVER-THE-AIR FIRMWARE UPDATE
[SOURCE: MACHNATION, 2020]



As shown on Figure 9-4 and confirming our prior statements, MQTT and LwM2M over HTTPS uses a small number of large packets to grab the firmware update file, whereas LwM2M over CoAP uses a larger number of small packets to complete the transfer. This difference may be important in environments where network connectivity is constrained by bandwidth or maximum MTU size, such as in low-power wide-area (LPWA) or 2G/3G cellular networks, where supported.

Power consumption: idle, 1, 30, and 60-second observation reporting intervals

MQTT devices consume 33% more power than LwM2M devices when measured at idle and 1, 30, and 60-second update intervals, when testing the AWS IoT Python-based SDK against the C-based Anjay SDK.

In this scenario, MachNation tested power consumption measured in watt-hours (Wh) of an IoT device. For MQTT, we tested the AWS IoT client connected to the AWS IoT platform, configured to send a device shadow update including one string, one floating-point number, one integer, and one Boolean value, at varying reporting intervals. For LwM2M, we tested the Anjay client connected to the AVSystem Coiote IoT Device Management platform with various server-initiated reporting intervals for a single custom LwM2M object containing the same set of data as the MQTT observation. The reporting intervals tested were

- Idle (no messages sent)
- 1-second interval (60 messages per minute)
- 30-second interval (2 messages per minute)
- 60-second interval (1 message per minute).

Overall, the measurement period was 10 minutes per tested reporting interval.

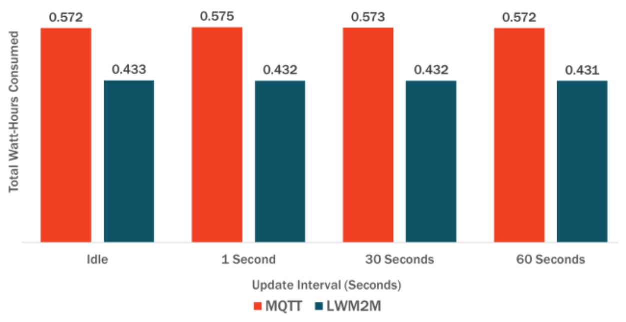
MachNation chose to test power consumption, because many IoT devices are battery powered and expected to last 10 or more years in the field. As such, any additional power consumption caused by the messaging or device management client can negatively impact device longevity.

Overall, MachNation found that a device with an MQTT client consumes 33% more power than a device with LwM2M. During the scenario, LwM2M and MQTT in the idle state consumed 0.572 and 0.433 Wh, respectively. In addition there is little significant difference across intervals tested (i.e., idle, 1 second, 30 second, and 60 second intervals). This is likely due to the lack of power management optimization in the Raspberry Pi hardware used for MachNation's tests, however, this finding will require additional testing to draw further conclusions.

Although it is tempting to attribute the LwM2M and MQTT power-consumption differences to the underlying protocols leveraged, MachNation suspects the observed differences may be due to the different device client frameworks implemented. Specifically, the Python-based AWS device client, while quite performant, requires extra computational effort to execute compared to the C-based Anjay client. The Anjay client is available only as a C-based library, whereas the AWS IoT SDK is available in a variety of programming languages, many of them based on higher-level languages such as JavaScript/Node.js or Python. While we attempted to test the AWS IoT SDK for embedded C, the current generally-available (GA) version of the embedded-C SDK lacks typical features compared to Amazon AWS' other GA SDKs. The replacement embedded-C SDK was not GA and as a result was not used for testing.

Below we present the data from the power-consumption scenarios.

FIGURE 10-1: POWER CONSUMPTION OF IOT DEVICE IN THE IDLE STATE AND UNDERGOING 1-SECOND, 30-SECOND, AND 60-SECOND UPDATE INTERVALS IN WATT-HOURS [SOURCE: MACHNATION, 2020]



As shown on Figure 10-1, during all four tests, MQTT consumed approximately 33% more power in watt-hours than LwM2M. Interestingly, the data update interval—whether 1, 30, or 60 seconds—did not impact the percentage difference in power consumption between MQTT and LwM2M. It is likely that a lack of power optimization in the System on a Chip (SoC) of the tested hardware as well as differences in the Python- and C-based SDKs contributed to the differences more so than any differences in the underlying communication protocols. In addition, while not tested here, the LwM2M specification does provide additional options for deep sleep between messaging intervals, while similar functionality must be custom implemented for comparable MQTT-based solutions.

Qualitative evaluation: business comparison

MQTT is more readily available, easier to procure, and easier to test than LwM2M.

There were several noteworthy differences in terms of ease of implementation from a business perspective when comparing the AWS and AVSystem device clients. Of particular note is the relative availability and ease of procurement of MQTT versus LwM2M clients and tools, even in cases of vendor-specific MQTT clients such as the AWS IoT SDK.

There are numerous MQTT libraries, implementations, and sample code available to help enterprises and developers rapidly deploy MQTT-based solutions. Amazon offers several versions of the AWS IoT SDK, all of them open source and supporting a wide variety of programming languages. Analogous solutions can be found for other public-cloud vendors such as Microsoft Azure IoT. Of note, nearly all IoT platform vendors release the full source code for multiple languages for their respective MQTT implementations. In general, there are many projects and applications within the IoT world that support MQTT, with it largely being viewed as a *de facto* standard for IoT devices.

Conversely, there are only a few reference implementations for LwM2M version 1.0 currently available on GitHub, although some chipset and module manufacturers provide their own LwM2M clients for their products.[18] The relative lack of reference LwM2M clients is problematic for an ecosystem that desires to increase LwM2M's market share relative to MQTT. As mentioned, MachNation recommends that the LwM2M ecosystem creates productized, publicly available, reference LwM2M clients with supporting documentation to empower enterprise developers to create IoT solutions and POCs using LwM2M as easily as with MQTT.

Qualitative evaluation: technical comparison

Initial implementation complexity associated with LwM2M can yield more cost-effective, technically-flexible IoT deployments compared to MQTT.

There are many positives for the LwM2M relative to the MQTT protocol in terms of technical capabilities. As already summarized above, MachNation found LwM2M to be more efficient in terms of packets and bytes sent and received during testing. LwM2M also has distinct advantages around the predictability of packet sizes, with more consistent packet exchanges during routine operations such as firmware updates as well as during observation reporting. LwM2M benefits from its use of CoAP and UDP as the underlying transport compared to MQTT and TCP. TCP connections, by their nature, require an acknowledgement packet to be sent for every data packet sent. Additionally, some vendor-specific implementations, such as AWS' IoT Device Shadow service, are significantly less efficient in terms of redundant messaging compared to LwM2M, resulting in more chatty MQTT-based communication.

Although the total savings in terms of bytes transferred is significant for L2M2M and might yield cost savings from use of less data and lower power consumption, what is perhaps more significant is the predictability of LwM2M messages. With consistent packet sizes and message size per operation, both customers and network operators can readily optimize their communication layers to handle large numbers of connected IoT assets, especially over MTU-constrained networks. Additionally, while MachNation did not test these capabilities within this study, LwM2M offers several additional communication-layer benefits, such as non-IP data delivery (NIDD), enabling messages to be transferred over wireless control plane channels rather than traditional IP-based channels.

[18] All major cellular IoT hardware manufacturers including chipset, module, and end-device manufacturers have completed or initiated interoperability projects with AT&T. This suggests a significant increase in support for LwM2M in the telecommunications industry.

From a technical implementation perspective, MQTT tends to be faster to initially implement than LwM2M, although MQTT can negatively impact enterprises due to future, unanticipated development costs associated with vendor-specific implementations. MQTT is similar to traditional RESTful HTTPS-based interfaces, where the customer and platform are freely allowed to implement message and data schemas as desired. With MQTT, a developer can define a JSON object with desired attributes and freely push messages to and from the server and IoT devices. However, many vendors' products, such as Amazon AWS IoT and Microsoft Azure IoT, implement vendor-specific reserved topics and other vendor-specific authentication schemas for MQTT. These schemas make switching IoT platforms more difficult, because switching would require a developer to refactor code, even if the underlying MQTT protocol used on the devices is the same.

LwM2M is more difficult to implement correctly today.[19] Achieving the same result with LwM2M as with MQTT requires the device client to normalize data before it is sent to the broker. It also requires the developer to potentially implement custom object types that require separate out-of-band coordination of XML-type definitions. Further, the complexities of load-balancing and scaling UDP connections carrying DTLS-encrypted CoAP packets can add additional complexity for IoT platform vendors with large device deployments.

However, the upside to all the additional LwM2M complexity is a well-standardized, cross-vendor protocol that can save developers time over the life of an IoT deployment. Rather than letting the vendor define various reserved topics or vendor-specific authentication schemas, LwM2M provides a well-defined schema for both device management and observation data delivery. Leveraging OMA's IPSO Smart Objects to provide schemas for many data types, LwM2M enables any device with a well-defined observation type to report that data in a non-vendor-specific dialect. Further, LwM2M provides standard methods for firmware updates, management of observation reporting intervals, and many typical device management functionalities. All of this is available in the specification and while it can be challenging to implement[20] for customers new to IoT, the specification forces developers to resolve both immediate and future technical challenges at the inception of device integration, rather than allowing poor initial decisions to become apparent later in the deployment cycle.

Ultimately, LwM2M is a technical specification, which may, given proper enablement from platform vendors, ultimately supplant the fragmented MQTT landscape as the *de facto* standard for IoT devices. For specific use cases, particularly those using cellular and constrained/low-powered devices, it is possible that most of the complexity of LwM2M could be handled by chipset and module manufacturers that choose AT commands as the primary method of communication with the underlying LwM2M library and are willing to offload device management and communications to a baseband chipset vendor.

[19] The difficulties in implementing LwM2M could arise from technology, documentation, or vendor-support issues. However, it is quite possible that as LwM2M matures, these deficiencies will be addressed.
[20] See footnote 19.

Areas of Future Suggested Research

As with all high-quality, primary, test lab-based research, there are more areas to investigate. MachNation suggests the following areas of future research to further clarify the benefits and shortcomings of LwM2M- and MQTT-based solutions:

- Testing of LwM2M in constrained-network environments, such as LPWAN networks including NB-IoT and LTE Cat M1, 2G/3G networks (where they exist and are relevant for customer implementations), and LwM2M over control-plane communication protocols
- Testing of power consumption on power-optimized and CPU-constrained devices, such as lower-powered Arm-based SoCs or MCUs
- Testing of additional LwM2M device client frameworks, such as the Eclipse Foundation's Leshan client SDK and possibly LwM2M clients from various chipset and module manufacturers, some of which are mentioned in footnote 18
- Testing of additional vendor-specific MQTT implementations, such as Microsoft Azure IoT's device SDK

Conclusions

By choosing the right technology protocol, IoT developers will help enterprises bring their IoT solutions to market faster, save ongoing development and management costs, and future-proof their IoT solutions.

One of the most important technology protocols is the one that facilitates platform-to-device communications and, in some cases, supports management of IoT devices. LwM2M and MQTT are two of the most common protocols that enterprise IoT developers consider to solve these challenges.

To help developers and enterprises make fact-based choices when selecting technology, MachNation, with technology support from AT&T and AVSystem, designed and completed a set of hands-on tests to investigate the relative efficiencies of LwM2M and MQTT protocols.

In summary, this research found that on a typical IoT device using an MQTT client versus one using a LwM2M client:

- LwM2M shows efficiency and performance benefits over MQTT in almost all test categories including amounts of data transferred during the initial device-to-platform connection (or after device reboot), the steady ongoing state of a device connection, device observations at 2 updates per minute, and a single platform-to-device message push. In addition, a LwM2M-equipped device consumes less power than a similarly-equipped MQTT device irrespective of the update interval, although it is possible that such differences stem from the device client framework rather than protocol used. Finally, LwM2M requires more forethought during the design and development process[21], but yields several important technical advantages.

- MQTT shows slight[22] efficiency and performance benefits over LwM2M in the amount of data transferred during an OTA firmware update. MQTT is easier and faster to deploy on an IoT device[23], though allows many important device-integration questions to go unresolved during initial deployment and rollout, possibly creating unknown, future costs for enterprises.

MachNation will continue to investigate the differences between IoT technologies like LwM2M and MQTT in subsequent work.

[21] The difficulties of implementing LwM2M clients likely come from LwM2M's relatively sparse documentation and lack of support for existing open-ecosystem clients relative to MQTT.
[22] As previously discussed, the slight efficiency benefits of MQTT over LwM2M could be insignificant. MachNation supports the results of this whitepaper based on the technology selected for testing as shown on Figure 3. MachNation also recommends additional testing of OTA updates using different clients and devices.
[23] This relative ease of deployment likely comes from ease of implementation on both server- and client-side technology, good documentation, and strong support for existing open-ecosystem clients.

About Project Sponsors

MachNation would like to thank AT&T and AVSystem for their sponsorship of this technical study.

About AT&T Communications

We help family, friends and neighbors connect in meaningful ways every day. From the first phone call 140+ years ago to mobile video streaming, we innovate to improve lives. We have the nation's fastest wireless network.** And according to America's biggest test, we have the nation's best wireless network.*** We're building FirstNet just for first responders and creating next-generation mobile 5G. With a range of TV and video products, we deliver entertainment people love to talk about. Our smart, highly secure solutions serve nearly 3 million global businesses – nearly all of the Fortune 1000. And worldwide, our spirit of service drives employees to give back to their communities.

AT&T Communications is part of AT&T Inc. (NYSE:T). Learn more at att.com/CommunicationsNews.

AT&T products and services are provided or offered by subsidiaries and affiliates of AT&T Inc. under the AT&T brand and not by AT&T Inc. Additional information about AT&T products and services is available at about.att.com. Follow our news on Twitter at @ATT, on Facebook at facebook.com/att and on YouTube at youtube.com/att.

**Based on analysis by Ookla® of Speedtest Intelligence® data average download speeds for Q3 2019. Ookla trademarks used under license and reprinted with permission.
***GWS OneScore, September 2019.

About AVSystem

No IoT deployment is successful without proper device management – this is what AVSystem stands for. We help companies around the globe deliver better quality of service thanks to our best-in-class device management solutions. We also focus on WiFi VAS and indoor location as well as systems for SDN and NFV. In addition to creating software, we actively participate in the standardization process of the Lightweight M2M (LwM2M) standard to enable secure device management in the IoT ecosystem. More than 100 companies worldwide, including some of the world's largest mobile network operators, rely on AVSystem to expedite their IoT service deployments. Learn more at www.avsystem.com.

About MachNation

MachNation is exclusively dedicated to testing and benchmarking Internet of Things (IoT) platforms, middleware, and services. MachNation owns and runs MachNation IoT Test Environment (MIT-E), the industry's only independent, hands-on, benchmarking lab for IoT platforms. As the first provider of IoT performance and scalability testing, MachNation testers, developers, and analysts provide guidance to industrial enterprises, the world's leading IT vendors, and communication service providers. MachNation participates in many of the world's most exclusive IoT events and contributes regularly to leading IoT and business press. For more information, contact us.