Extensions to LwM2M for Improved Efficiency

Matthias Van Eeghem, Abdulkadir Karaagac, Jen Rossey, Bart Moons, Eli De Poorter, Jeroen Hoebeke
Ghent University - imec, IDLab, Department of Information Technology
Ghent, Belgium
Email: abdulkadir.karaagac@ugent.be, jeroen.hoebeke@ugent.be

***Abstract*—Over the past few years, we have seen several research efforts and innovations targeting the extension of Internet technologies into constrained networks. These efforts resulted in several networking standards and protocols which has led to the emergence of the Internet of Things (IoT). One of these protocols is LwM2M, a standard for device and service management in Machine-to-Machine (M2M) communication. In this paper, we propose the Batch object, an extension to the LwM2M protocol, in order to improve communication efficiency in LwM2M. This object allows the user to perform actions on multiple resources in a device by sending a single request. It can also be used to enable a reverse interaction model in LwM2M, which allows periodic updates of multiple resources without any need for a request.**

***Index Terms*—IoT, CoAP, LwM2M, Efficiency, Batch**

I. Introduction

As the ideas and technologies behind the Internet of Things (IoT) take root, a vast array of new possibilities and applications will emerge with a significant number of devices being connected to the Internet. The idea is that everyday items (sensors, actuators, vehicles, wearables, medical implants etc.) are being equipped with the end-to-end Internet connectivity. It is estimated that the IoT will become a enormous ecosystem that will consist of around 30 billion connected devices by 2020 [1].

The majority of these intelligent *things* are envisioned to be constrained in terms of processing capability, memory and energy as well as to have low unit price and long battery life. Due to these limitations, widely-used Internet protocols are not appropriate for constrained devices. Therefore, a new set of open protocols and standards were designed to be used by these devices in order to integrate them with the rest of the Internet.

In this context, several standards were proposed and have been used in order to achieve energy and resource efficient communication for constrained devices. Under the lead of the Internet Engineering Task Force (IETF), many working groups have been formed for the realization of IP-based connectivity for constrained devices [2], [3], [4]. The combination and coordination of these protocols creates a standardized way for integrating these constrained devices into the Internet, which makes them an important enabler of the IoT. In addition to the IETF, several research and standardization initiatives are gathered and constituted to target interoperability and M2M understandability in the IoT. For instance, oneM2M [5], Open Mobile Alliance (OMA) [6] and the Internet Protocol for Smart Objects (IPSO) Alliance [7] are leading global organizations that deliver specifications and architecture for creating efficient Machine-to-Machine (M2M) communication and global interoperability for the future of the IoT.

However, there is still a long way to go regarding the efficiency of these protocol and standards. Therefore, in this paper, we try to improve the efficiency of the LwM2M IoT protocol. First of all, we add the ability to write to and read from a subset of resources of an object or multiple resources over different objects using a single request with a newly proposed object: the *Batch*. Secondly, the Batch object creates functionality in LwM2M to get periodic updates of multiple resources within a single notification packet, without any need for a request, which we call the *reverse interaction model*. Also a theoretical and practical analysis is provided on the proposed solution in order to demonstrate its potential to achieve more energy and resource efficient communication in the LwM2M IoT Protocol.

The remainder of the paper is organized as follows: section II provides a detailed background about the target technologies in order to understand the current state of the art. Section III discusses the related work. In section IV, the Batch object is introduced in details. Section V provides theoretical and practical analysis about the contribution of the Batch approach. Finally, section VI concludes the paper.

II. Background

A. The Constrained Application Protocol (CoAP)

CoAP [8] is a dedicated REST-based web transfer protocol for resource constrained nodes and networks, standardized by IETF CoRE working group [8]. Like HTTP, CoAP also relies on the REST-based principles. Due to its lower header overhead and lower parsing complexity, CoAP is much lighter than HTTP, which makes it suitable for applications running on top of constrained devices. [9]

B. The Lightweight Machine-to-Machine (LwM2M) Protocol

LwM2M is an efficient and secure client-server protocol, from the Open Mobile Alliance (OMA), with several functionalities for managing resource constrained IoT devices [10]. LwM2M defines efficient interactions for remote application management and the transfer of service and application data via several interfaces built on top of the CoAP protocol. The overview of the typical LwM2M interaction model and the structure of the LwM2M object model is presented in Figure 1.

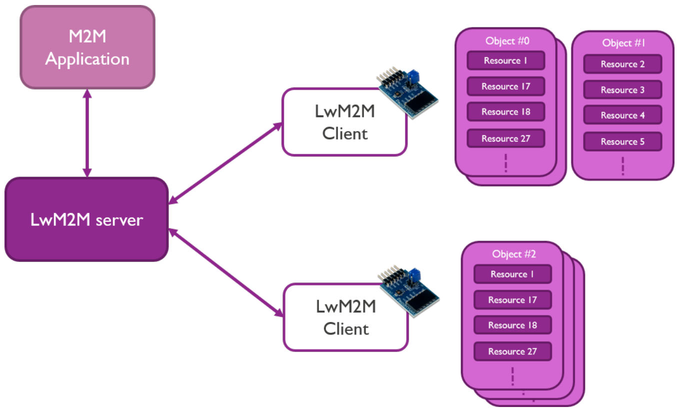

Fig. 1. An overview of LwM2M APIs and Data Models.

The LwM2M protocol relies on uniform object models which are collections of mandatory and optional resources representing atomic pieces of information. These object models are sets of pre-structured URI templates and registries of object identifiers with semantic attached. This provides machine-readable representations of the semantics.

LwM2M clients hold multiple objects, each with a unique identifier: the object ID. Some objects are mandatory (e.g. LwM2M Security) and some are optional. Each object can have zero or more instances. Every instance shares the same set of resources but the exact values in these resources can differ from instance to instance. Each resource within these objects has a Resource ID (RID), which uniquely defines the resource. Referencing a specific resource is done using the Uniform Resource Identifier (URI) */object_id/instance_id/resource_id*. Referencing an object is done using */object_id* and */object_id/instance_id* is used to reference a specific object instance. As an example, the LwM2M/IPSO GPS Location object is provided in Table I [11].

TABLE I
THE LWM2M/IPSO GPS LOCATION OBJECT (ID: 3336).

RID	Name	R/W	Type	Description
5514	Latitude	R	String	The decimal notation of latitude.
5515	Longitude	R	String	The decimal notation of longitude.
5516	Uncertainty	R	String	The position accuracy in meters.
5517	Velocity	R	Opaque	The device velocity.
5518	Timestamp	R	Time	The time of location measurement.
5705	Compass Direction	R	Float	Measured Direction.
5750	Application Type	R/W	String	Application type of the device.

As presented in Table I, the LwM2M standard allows to create resources with different access types: Read, Write, Read/Write, Execute. The LwM2M specification allows resources with a wide range of data types: String, Integer, Float, Boolean etc.[10]. It also allows homogeneous lists of these types but it is not possible to mix types within a single list. According to LwM2M these data types can be encoded in a number of supported data formats; TLV (Type-Length-Value), JSON, Plain Text and binary data formats.

III. RELATED WORK

In literature, there are a number of attempts which use a similar approach which can be used to improve the efficiency of the LwM2M protocol. Three alternative approaches are discussed below. The first is an idea built directly on top of LwM2M, the other two target the underlying CoAP protocol, however they are so far considered in the context of LwM2M.

IPSO defines the concept of a *Composite Object* where multiple objects can be combined using the Web Linking Framework [12]. For instance, a thermostat object can be created by aggregating Temperature, Setpoint and Actuation objects linked with input, setpoint and output links. However, this concept does not allow aggregating non-related (not linked) objects or a subset of resources within diverse objects. Therefore this approach is not flexible enough.

In [13], Van der Stok et al. has proposed three new CoAP methods (FETCH, PATCH and iPATCH), in addition to primary CoAP methods (GET, PUT, POST, DELETE), which can be used to access and update only a certain part of a resource. FETCH allows users to read only a part of an object instance. Whereas, PATCH and iPATCH can be used, unlike PUT, to partially update an object. These methods improve the efficiency compared to primary CoAP methods, however they do not allow operations on multiple resources across different objects. Moreover, these methods introduce a level of overhead due to added target resource URIs in the request.

In [14], Wang et al. propose a method utilizing multiple URIs where a M2M application could generate a Multiple URIs (MU) type request, encapsulating multiple resources. If a client receives a MU type request, it can generate a Multiple Value (MV) type response. Alternatively, an intermediate node or gateway can also aggregate multiple single URI (SU) type requests into one MU-type request. However, there is a certain overhead of having to add the URI for every resource to every request.

IV. THE BATCH

Although the LwM2M protocol is providing efficient interactions for transferring service and application data, there are still some limitations.

LwM2M allows reading an entire object instance or a certain resource by using a single request, however it is not possible to read a subset of resources of an object or multiple resources over different objects using a single request. Currently, separate requests have to be sent in order to read the values of multiple resources. Similarly, it only allows an object instance to be updated partially or entirely using a single request. This means that it is possible to edit multiple resources simultaneously as long as they belong to the same object instance. As is the case with reading, it is not possible to write to multiple resources across object instances using a single request. Another limitation of LwM2M is that it is not possible to get periodic updates of multiple resources in a single packet, without having to send requests. This is what the Batch attempts to solve using the reverse interaction model.

A. *The LwM2M Batch Object*

The Batch object is a new custom LwM2M object and was proposed by Karaagac et al. [15]. In addition to the mandatory resources (Batch Configuration, Batch Value), the Batch object also includes optional resources in order to create a reverse interaction model in LwM2M. The overall Batch object is presented in Table II with all created resources.

TABLE II
THE BATCH OBJECT (ID: 2346).

RID	Name	R/W	Type	Description
5913	Batch Configuration	R/W	String (List)	The list of aggregated resources.
5914	Batch Value	R/W	String (List)	Values of the resources defined in Batch configuration.
5915	SSID	R/W	Integer	Short Server ID.
5916	URI Path	R/W	String	URI path on the server.
5917	Periodicity	R/W	Integer	Period of uplink transmissions (sec).
5918	Confirmable	R/W	Boolean	Set CoAP messages to be Confirmable.
5919	No-Response	R/W	Boolean	Enable No-Response option in message.

The Batch Configuration (5913) resource allows users to specify the resources that they are interested in, whereas the Batch Value (5914) resource allows the user to read from or write to the resources defined in the Batch Configuration. Both of these resources have type of ’list of strings’ and are both readable and writable.

An example flow of the Batch creation, read and write operations is visualized in Figure 2. In this example, the Batch configuration is first set to contain two resources: */x/y/z* and */a/b/c*. The second step simply reads back what was just written. The configuration and value resources are directly linked with each other: the first string in the Batch value resource corresponds to the first URI in the Batch configuration etc. In the example, the value 42 corresponds to the resource */x/y/z*, and ”test message” is the value of resource */a/b/c*. In the third step, the Batch value resource is read. The values of the resources in the Batch configuration are collected into a list and returned. Those values are collected into a list of strings and returned in a single request. The fourth step shows the operation to edit the values of multiple underlying resources at a time.

Since each operation on batch object applies to each of the aggregated resources, Read-only and Write-only resources cannot be aggregated into a single batch. Therefore, there needs to be a validation process for the content of the batch during Batch creation.

When using the Batch object, updating multiple resources is an atomic operation. Either all resources of the batch are updated or none are, e.g. in case the packet is lost for example. Using individual requests to update multiple resources in or across objects, the data in the resources might be partially updated in case of packet loss and therefore left in an invalid state. With Batch, one can ensure that all necessary resources are updated jointly which will and the object is not left in an invalid state.

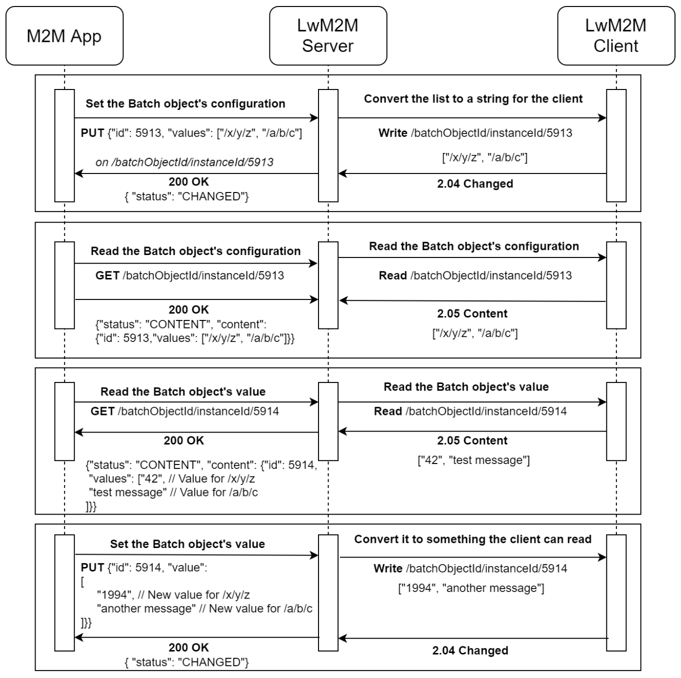
M2M App
LwM2M Server
LwM2M Client
Set the Batch object's configuration
PUT {"id": 5913, "values": ["/x/y/z", "/a/b/c"]
on /batchObjectId/instanceId/5913
200 OK
{ "status": "CHANGED"}
Convert the list to a string for the client
Write /batchObjectId/instanceId/5913
["/x/y/z", "/a/b/c"]
2.04 Changed
Read the Batch object's configuration
GET /batchObjectId/instanceId/5913
200 OK
{"status": "CONTENT", "content":
{"id": 5913,"values": ["/x/y/z", "/a/b/c"]}}
Read the Batch object's configuration
Read /batchObjectId/instanceId/5913
2.05 Content
["/x/y/z", "/a/b/c"]
Read the Batch object's value
GET /batchObjectId/instanceId/5914
200 OK
{"status": "CONTENT", "content": {"id": 5914,
"values": ["42", // Value for /x/y/z
"test message" // Value for /a/b/c
]}}
Read the Batch object's value
Read /batchObjectId/instanceId/5914
2.05 Content
["42", "test message"]
Set the Batch object's value
PUT {"id": 5914, "value":
[
"1994", // New value for /x/y/z
"another message" // New value for /a/b/c
]}}
200 OK
{ "status": "CHANGED"}
Convert it to something the client can read
Write /batchObjectId/instanceId/5914
["1994", "another message"]
2.04 Changed

Fig. 2. A sequence diagram showing the interaction with the Batch object.

The Batch object also supports CoAP Observe functionality, which allows CoAP clients to observe resources on CoAP servers. If a value of a resource is changed through the Batch object, the Batch object makes sure to propagate this change to all of the potential observers of this resource. The reverse is done as well: if a resource is present in the Batch configuration and it is changed externally (i.e. not through the Batch object), observers of the Batch object are notified of a change as well.

B. *Reverse Interaction via the Batch*

In addition to resource aggregation, the Batch object is also offering functionality to create reverse interaction models, which is normally not supported by the LwM2M protocol. This feature allows a user/server to register on a device to periodically receive updates for a Batch of resources. Therefore, the user no longer has to send requests for periodical updates.

In order to activate this reverse interaction model, a number of additional resources defined in the Batch object are being used. Firstly, *SSID* represents the resource defining which external server that the update should be sent to. Secondly, *URI Path* is the exact path on the server to write the update to using a PUT operation. The payload of this update message is the value of the Batch value resource with resource ID 5914. Further, *periodicity* defines how often an update should be sent expressed in seconds. The resource *Send Confirmable Message* lets the user define which type (confirmable, non-confirmable) of CoAP message should be sent. Finally, *No Response Option* is used to make the server not send a response to specific (or all) types of messages as described in [16].

V. Evaluation

In this section, the efficiency of the Batch object is evaluated via theoretical and practical analysis.

A. Theoretical Analysis

In order to investigate the potential contribution of the Batch approach, we performed an analysis of various communication scenarios involving LwM2M devices. On the one hand, we calculated the communication performance in case of using multiple CoAP requests targeting different resources. On the other hand, we performed the same calculation for the case where a number of resources are aggregated in a single Batch object. The combination of these two calculations shows the contribution of the Batch approach in different scenarios and settings.

Initially, we analyzed the required data exchange in order to perform a Read operation on multiple LwM2M resources. As it can be expected, the Batch approach is resulting in aggregated responses with larger packet sizes. However, due to the smaller number of requests and responses, the Batch object provides a lot more efficient read operations in terms of the total number of bytes to be exchanged. This performance improvement can be seen in Figure 3 for using TLV and JSON encoding schemes. This figure proves the fact that the Batch operation is more efficient than using multiple requests in terms of bytes required to complete the request(s). For this experiment, we assumed to have a payload of 4 Bytes long.

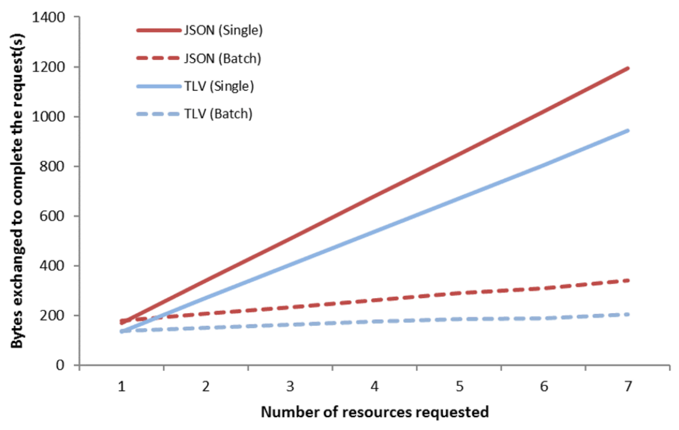

Fig. 3. The amount of bytes required to complete multiple requests as with the batch object or using single requests.

Secondly, we performed analysis about the impact of the Batch approach in terms of latency. For this purpose, we investigated the use of the Batch operation in a bandwidth-constrained Low Power Wide Area Network (LPWAN), using LoRa as a suitable example [17].

To calculate the latency in a LoRa network, Equation 1 can be used. In this equation, the airtime represents the packet transmission time and it depends mostly on the spreading factor, coding rate and bandwidth used. LoRa uses spreading factors (SF7-SF12) to set the modulation rate. SF7 provides the shortest airtime and resulting range as well as the highest bit rate, whilst SF12 uses the longest airtime with the longest range and the smallest bit rate [18]. The second part represents the delay which would come from the duty cycle regulations and it is directly proportional to packet airtime. For instance, the duty-cycle is 1% in EU 868 for end-devices which means that an end-device can occupy a channel with a maximum 1% of the time. The constant (C) part consists of the processing delay and the time to transfer packets between the gateway and the cloud.

$$latency = airtime + D(airtime) + C \tag{1}$$

Considering a number of resources, the total latency in standard LwM2M approach would result in the total latency of all requests and responses, which is provided in Equation 2.

$$latency = \sum_{i=1}^{n} airtime(req_n) + airtime(resp_n) + D(a_n) + C \tag{2}$$

The Batch operation would require only one request which will provide a total latency which can be calculated by using Equation 3.

$$latency = airtime(req) + airtime(resp) + D(a) + C \tag{3}$$

The impact of the batch object over single requests, can easily be calculated using the information from the SX1276 datasheet [19]. For example, 1 single request consists of 16 bytes CoAP header and 11 bytes payload, which adds up to 27 bytes packet length, resulting in 82 ms of airtime. Requesting 6 resources separately would thus result in 492 ms of sending time. Aggregating 6 of these requests in a single Batch, would require a single 16 bytes CoAP header and 66 bytes of payload. These 82 bytes result in 164 ms of airtime, or 3 times less airtime for the same amount of data.

These equations clearly show that the Batch will improve the total latency of LwM2M communications. Initially, the Batch object decreases the number of request and response messages, so it decreases the total airtime of the LoRa packets. Secondly, as the number of packets decreases the delay, which comes from the duty cycle regulation, will also decrease drastically. Finally, requesting multiple single requests will add up the constant delay, whereas it will not change for the Batch operation.

However, combining multiple resources will not always result in a gain in efficiency. Whenever the size of the aggregated response message is larger than the maximum transfer unit (MTU) allowed to be sent over the network, the message will be split into multiple packets due to CoAP block transfer. This is expected to impact the communication performance negatively. This now poses a trade-off: using single requests requires a separate packet to be sent for every resource, but from a certain number of resources, the Batch object's packet size will exceed the MTU and it will also require multiple packets using block transfer. This second option could cause the Batch object to become less efficient.

B. Practical Evaluation: Exposing Device Location

In order to see how well this idea stacked up against the state of the art, the Batch object and related functionalities are implemented for both a LwM2M client and server. For the LwM2M client, the open source C implementation of Anjay was used and extended [20]. Anjay was extended with the proposed Batch object and the reverse interaction model. This object model was also implemented in a LwM2M server: Leshan [21]. It is also an open source solution and is implemented in Java.

In this evaluation, an external system (named Localization Server) calculates the position of a device and notifies the device of its new position. The device has the Localization Server registered as a server, which allows the Localization Server to directly send CoAP packets to the sensor instead of having to go through the LwM2M server. The position is always sent as a triple. New values are stored in the Position object, which can be seen in Table III.

TABLE III
THE LwM2M POSITION OBJECT (ID: 3360).

RID	Name	R/W	Type	Description
5701	Sensor Units	R/W	String	Measurement Units Definition.
5702	X Value	R/W	Float	The measured value along X axis.
5703	Y Value	R/W	Float	The measured value along Y axis.
5704	Z Value	R/W	Float	The measured value along Z axis.
5516	Uncertainty	R/W	Float	The accuracy of the position.
5518	Timestamp	R/W	Time	The time of location measurement.
5750	Application Type	R/W	String	The Application Type.

The resources with IDs 5702, 5703 and 5704 are updated every time a new position triple is received. For this purpose, two different approaches are evaluated. In the first, the new position is sent over three separate packets (x,y,z). The second, when using the Batch object, follows the following sequence:

1) the LwM2M client registers with the Localization Server.
2) a Position object instance is created. The instance ID is picked by the LwM2M client.
3) a new Batch object instance is created and the configuration is immediately set to the corresponding resource IDs in the Position object.

In each approach, the position is updated 10 times. The packets for creating the Position object and creating the Batch object are included in the calculations. The impact of having to create these instances in the first solution becomes smaller over time. The measured data traffic values are provided in Table IV. These values show that Batch approach provides a lot more efficiency than updating the (x, y, z) coordinates separately in three different requests.

C. Reverse Interaction Model

In this experiment, an external system wants to know the temperature (/3303/0/5700), humidity (/3304/0/5700), coordinates (X,Y,Z: /3360/0/5702, /3360/0/5703, /3360/0/5704) and battery state (/3/0/9) of a device with a period of 15 minutes.

TABLE IV
DATA TRAFFIC FOR DIFFERENT POSITION UPDATING POLICIES.

	Batch	**Three req.**
Bytes exchanged	2.065	4.203
Bytes sent	1.403	2.522
Bytes received	662	1.681
Payload sent (in bytes)	899	1.220
Payload received (in bytes)	158	379
Packets exchanged	24	62

For this purpose, three different approaches are considered. In the first approach, the external system makes six separate requests to the resources every 15 minutes, while the second approach uses the Batch object and the external system sends a single request for the Batch object's value every 15 minutes. The last approach is based on the reverse interaction model which automatically sends updates every 15 minutes to the external system.

The results can be seen in Table V. The results are calculated for 96 updates in total (96 * 15 minutes is one day) and the JSON data format is used. These results show that the Batch is almost six times more efficient than making six separate requests. The reverse interaction model is improving the efficiency even more by sending the data automatically to the registered server.

TABLE V
RESULTS FOR GETTING RESOURCE VALUES OVER MULTIPLE OBJECTS.

	Sep. requests	**Batch**	**Reverse Batch**
Bytes exchanged	95.904	30.259	23.658
Bytes sent	39.168	6.774	173
Bytes received	56.736	23.485	23.458
Payload sent (in bytes)	14.976	2.700	131
Payload received (in bytes)	32.544	19.411	19.411
Packets exchanged	1.152	194	98

D. To BATCH or Not To BATCH

The previous sections show that the Batch object can improve the communication efficiency in LwM2M drastically, however there might be cases or scenarios where the Batch approach does not improve the performance, even decrease in certain cases. Therefore, there are certain factors which should be taken into consideration while deciding whether or not to use Batch. This section summarizes these factors.

Firstly, the number of aggregated resources and their types are crucial factors which define the contribution of the Batch operation. If the total payload size of the aggregated response message exceeds the MTU and block transfer is used, the performance of Batch operation decreases.

Secondly, in case of observing a Batch object, with every resource in the Batch configuration (even those unchanged) being included in every notification, the packets containing the updated values are larger than needed. This means that, depending on how many resources the CoAP client wants to observe, how often each resource provides an update and how long the continued interest has to be shown, the CoAP client should thus choose between observing multiple resources at a time or observing a Batch object instance.

In addition, security should also be taken into account. Due to the significant overhead added by security protocols, the frequency of exceeding the MTU and having to use block transfer might grow drastically and would decrease the efficiency of Batch operation. Of course, security will also impact individual requests and this trade-off should be studied further.

Something that also has an impact on the decision in using either the Batch object or single requests is packet loss. Using the Batch object requires only a single response, while using multiple requests to get values requires multiple responses. That means, in case of packet loss, all aggregated data will be lost.

VI. Conclusion

The main advantage of using the Batch object is the decreased number of packets and total bytes that have to be sent and received. Instead of having to issue multiple requests for reading from and writing to multiple resources, the Batch object allows these operations to be done using a single request. Despite the larger packet size of the resulting packet, the Batch approach improves the communication efficiency in terms of data traffic and total latency due to a decrease in the number of packets to be exchanged.

The Reverse Interaction model is a useful extension on top of the Batch object. When an external system wants to periodically poll resources, it makes use of reverse interaction model. Instead of the system having to send a request every so often, the Batch object sends these updates automatically. This increases efficiency even more.

However, there are certain factors that should be taken into account in the decision process of the Batch usage: number of resources, resource types, encoding type, update rates, resource data size and the MTU in network. Such factors affect the efficiency of the Batch operation and the unthoughtful usage of Batch object can lead to performance drops.

For future work, a more efficient encoding for the value resource of the Batch object can be studied. The value resource is a list of strings at the moment, which is not efficient if the resources are integers, floating point numbers or booleans. Also a way to encode the arrays and object links should be further investigated.

Acknowledgment

Part of this research was funded by the Flemish FWO SBO S004017N IDEAL-IoT (Intelligent DEnse And Longe range IoT networks) project.

References

[1] A. Nordrum, “Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated,” August 2016.
[2] “IPv6 over Low power WPAN (6LoWPAN).” http://datatracker.ietf.org/wg/6lowpan, accessed on January 31, 2018.
[3] “Routing Over Low power and Lossy networks (ROLL).” http://datatracker.ietf.org/wg/roll, accessed on January 31, 2018.
[4] “Constrained RESTful Environments (CORE).” http://datatracker.ietf.org/wg/core/, accessed on January 31, 2018.
[5] oneM2M, “White Paper: The interoperability enabler for the entire M2M and IoT ecosystem,” January 2015.
[6] Open Mobile Alliance. http://openmobilealliance.org/, accessed on January 31, 2018.
[7] Internet Protocol for Smart Objects (IPSO) Alliance. https://www.ipso-alliance.org/, accessed on January 31, 2018.
[8] Z. Shelby, K. Hartke and C. Bormann, “The Constrained Application Protocol (CoAP),” RFC 7252, IETF, June 2014.
[9] I. Ishaq, D. Carels, G. Teklemariam, J. Hoebeke, F. Abeele, E. Poorter, I. Moerman, and P. Demeester, “IETF Standardization in the Field of the Internet of Things (IoT): A Survey,” *Journal of Sensor and Actuator Networks*, vol. 2, p. 235287, Apr 2013.
[10] Open Mobile Alliance, “Lightweight Machine to Machine Technical Specification,” February 2017.
[11] Open Mobile Alliance, “OMA LightweightM2M (LwM2M) Object and Resource Registry.”
[12] J. Jimenez, M. Koster and H. Tschofenig, “IPSO Smart Objects,” January 2016.
[13] P. van der Stok, C. Bormann and A. Sehgal, “PATCH and FETCH Methods for the Constrained Application Protocol (CoAP),” April 2017.
[14] C. Wang, Dale N. Seed, et. al, “Method and apparatus for using multiple universal resource identifiers in M2M communications,” August 2013.
[15] A. Karaagac, F. Van Den Abeele, J. Hoebeke, “Challenges for Semantic LwM2M Interoperability in Complex IoT systems.,” July 2017.
[16] A. Bhattacharyya and S. Bandyopadhyay and A. Pal and T. Bose, “Constrained Application Protocol (CoAP) Option for No Server Response,” August 2016.
[17] N. Sornin, M. Luis, T. Eirich, T. Kramp, O. Hersent, “LoRa specification. Technical report,” January 2015.
[18] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, “A Study of LoRa: Long Range and Low Power Networks for the Internet of Things.,” *Sensors*, September 2016.
[19] Semtech Corporation, *SX1276/77/78/79 datasheet*, August 2016. https://www.semtech.com/uploads/documents/sx1276.pdf, accessed on January 31, 2018.
[20] AVSystem, “Anjay, Open-source LwM2M Library.” https://www.avsystem.com/products/anjay/, accessed on January 31, 2018.
[21] Eclipse, “Eclipse Leshan is an OMA Lightweight M2M (LWM2M) implementation in Java..” https://eclipse.org/leshan/, accessed on January 31, 2018.