Aggregated Sensor Payload Submission Model for Token-Based Access Control in the Web of Things

Mohammad Amir, Prashant Pillai and Yim-Fun Hu
School of Engineering and Informatics,
University of Bradford, Bradford, UK
Mamir3@bradford.ac.uk, P.Pillai@bradford.ac.uk, Y.F.Hu@bradford.ac.uk

***Abstract*—Web of Things (WoT) can be considered as a merger of newly emerging paradigms of Internet of Things (IoT) and cloud computing. Rapidly varying, highly volatile and heterogeneous data traffic is a characteristic of the WoT. Hence, the capture, processing, storage and exchange of huge volumes of data is a key requirement in this environment. The crucial resources in the WoT are the sensing devices and the sensing data. Consequently, access control mechanisms employed in this highly dynamic and demanding environment need to be enhanced so as to reduce the end-to-end latency for capturing and exchanging data pertaining to these underlying resources. While there are many previous studies comparing the advantages and disadvantages of access control mechanisms at the algorithm level, vary few of these provide any detailed comparison the performance of these access control mechanisms when used for different data handling procedures in the context of data capture, processing and storage. This study builds on previous work on token-based access control mechanisms and presents a comparison of two different approaches used for handling sensing devices and data in the WoT. It is shown that the aggregated data submission approach is around 700% more efficient than the serial payload submission procedure in reducing the round-trip response time.**

Keywords—Internet of Things; Web of Things; Access control; Data Aggregation; Big Data.

I. Introduction

The Internet of Things (IoT) refers to a network of internet-enabled devices which can be accessed and interacted with via the internet. The Web of Things (WoT) is an extension of the IoT and focuses more on the web-based representation and interaction of internet-enabled devices (or “things”) [1]. The WoT enables virtual representation of devices and their related assets on the World Wide Web. The virtual representation of devices and their data opens up a plethora of opportunities for the WoT since digital devices can now be as easily browsed, indexed, and interacted with as traditional web pages [2]. Examples of such opportunities include the ability to use HTTP verbs in a Representational State Transfer (REST)-ful architecture to virtually poll, monitor and control physical devices. The representations of these devices are commonly referred to as *resources* [3].

The WoT is synonymous with huge data traffics and highly volatile and rapidly changing data. This makes traditional access control mechanisms such as User-Based, Authorization-Based and Role-Based Access Control (i.e. UBAC, ABAC and RBAC respectively) highly unsuitable. Instead, a more flexible and resource-oriented access control mechanism is required [4]. It has been shown in a previous study that Token-Based Access Control (TBAC) mechanisms combined with a RESTful Application Programming Interface (API) architecture are highly appropriate for handling data in the WoT [5]. A novel approach, Cascading Permissions Policy Model (CPPM), was used to provide efficient scalability of the TBAC mechanism for the WoT [5].

This paper builds on this earlier study and proposes a new aggregated CPPM-TBAC model for submitting sensor payloads to the server. Sensor payloads are packages containing either sensor definitions (e.g. sensor ID, name, description, properties, etc.) or sensing data (e.g. timestamp, sensor reading, etc.), and are described more thoroughly in subsequent sections. The server is a host machine which processes the submitted payloads, stores them in a database and uses the data in subsequent knowledge generation processes. Again, more details are contained in the following sections. The paper focuses on comparing the previously defined serial sensor payload submission model and procedure against the newly proposed aggregated sensor payload submission model and procedure. The aim of the study is to identify the most efficient approach which has the smallest possible round-trip response time so that the most suitable and appropriate scheme can be employed for the highly volatile WoT environment. Since the access control mechanism is present and employed in each submission of a sensor payload, it is paramount that a highly efficient submission model and procedure is devised in order to minimise delays and maximise the network efficiency in terms of handling more payloads in lesser time.

The rest of this paper is organized as follows: Section II discusses the need for access control and briefly outlines the CPPM-TBAC mechanism. Section III goes into more detail regarding the CPPM-TBAC mechanism and describes the model and procedure for the previously defined serial sensor payload submission mechanism and the bigger semantic framework which the CPPM-TBAC mechanism forms a part of. In Section IV, the newly proposed aggregated sensor payload submission procedure is described and compared against its predecessor. This is followed by an in-depth performance evaluation in section V, showing the improved efficiency of the aggregated sensor payload submission approach and its suitability for reducing the round-trip response time when interacting with devices on the WoT. Finally, Section VI presents the conclusions of the paper.

II. CPPM-TBAC

In the WoT, capturing and processing an unbounded number of devices (sensors, actuators, virtual entities, etc.) is a reality [6]. These resources are typically very temporal and short-lived which leads to dynamic and unpredictable application scenarios and interaction patterns [7]. In short, the following characteristics of cloud-based WoT repositories can be concluded:

- Unbounded: New resources (both physical and virtual) can be introduced at any time. For example, new devices may be introduced as more equipment becomes available at a disaster scene.
- Temporal: Resources are generally short-lived and undergo various changes in their properties and definitions. For example, legacy or faulty devices will be replaced with newer or more capable platforms over time. Also, the repositories may only store a certain amount of historical data and any data outside this boundary will become unavailable.
- Dynamic: Resources, their properties and definitions can change dynamically in response to events or over time. For example, a monitoring event in a natural disaster may cause several devices in the near vicinity to activate automatically.

Furthermore, for the WoT to truly flourish and be deployed in a useful context, accessing resources should be easy, intuitive and hassle-free. At the same time, access to private resources should be protected and the means of accessing this data should not be very complex and unintuitive so as to hinder user adoption.

The main purpose of an access control mechanism is to limit access to privately-owned resources and assets by the owner of these resources. In this regard, several methodologies exist:

1. User/Identity-Based Access Control (UBAC)
2. Authorisation-Based Access Control (ABAC)
3. Role-Based Access Control (RBAC)
4. Token-Based Access Control (TBAC)

In a previous study, the advantage of using TBAC over the other access control mechanisms for the data handling and processing needs of the WoT has been clearly identified [5]. TBAC systems are based on the premise of reusable and reconfigurable tokens that grant access to a set or group of protected/private resources for a particular user [8]. After generation, the tokens are transmitted to users/agents who need to consume private resources. These private resources are hidden from public view by default and are accessible only by the resource owner. Tokens can be configured to only expose the required resources and assets without exposing the identity of the resource owner. This is advantageous over UBAC which requires the identity of the user to be transmitted with each request to access protected resources. While roles in RBAC are a part of the overall organizational structure and are therefore more permanent and long-term artefacts, tokens in TBAC are much more decoupled since they are resource-oriented and can be easily generated, modified and revoked without affecting the organization structure. This provides a significant managerial advantage when tokens are used to control access to temporal assets of the network. Finally, since tokens are tied to resources as opposed to users who own those resources, this scheme provides a resource-centric access control scheme which is suitable for managing interactions with resources in a WoT setting.

The CPPM-TBAC is part of a larger semantic collaboration framework known as SAW: Semantically-enriched and Semi-autonomous collaboration framework for the WoT [9]. The CPPM-TBAC works over the asset model for SAW which represents resources at different levels of granularity and expressiveness. By utilising a RESTful API, resources are exposed as web-accessible URIs (Uniform Resource Identifiers) which can be interacted with using the 4 common HTTP verbs: POST for creating, GET for querying, PUT for updating and DELETE for removing resources [10]. The performance of the CPPM-TBAC in the context of serial sensor payload submissions has already been detailed previously [5].

This paper extends the existing work by proposing a new model and procedure for the CPPM-TBAC to support aggregated sensor payload submissions. The performance of the newly proposed mechanism will be evaluated in detail and compared against the previously defined serial sensor payload submission procedure. The consequent sections present the methodologies of the two different procedures as well as a critical numerical analysis to determine which procedure fares better in terms of the round-trip response time.

III. SAW FRAMRWORK WITH SERIAL SENSOR PAYLOAD SUBMISSION PROCEDURES

This section provides a brief description of the asset model of the SAW framework [9] in regards to the terminology used in the rest of the paper. SAW has a simple but extensible data hierarchy as illustrated in Fig. 1. A *datafeed* (DF or *feed*) implements a generic device template which can be used to model and represent any kind of physical or virtual device within a specific environment, for example, an Arduino board or a twitter user respectively. A *feed* has one or more *datastreams* (DS or *stream*) that describe a particular sensor or actuator asset of the *feed*, for example, a light sensor on an Arduino board or a twitter user's tweet stream. Finally a *stream* can have zero or more *datapoints* (DP or *point*), where each *point* references a particular value at a given instance in time, for example, a time-stamped light sensor value or a particular tweet from the stream of a twitter user.

This asset model enables modelling of sensing devices in any environment or at any level of granularity, using the generic and extensible data definition templates adopted to describe the assets. The CPPM-TBAC controls access to resources in this asset model starting from the most verbose, expressive and comprehensive datafeeds right down to the least expressive and cardinal datapoints. Tokens effectively enable the modelling of multi-faceted and cascading sets of permissions for accessing resources on the network. A set of tokens are generated automatically for each datafeed to represent a common set of read and write permissions. Further tokens can be generated by users for refining access to datafeeds and datastreams.

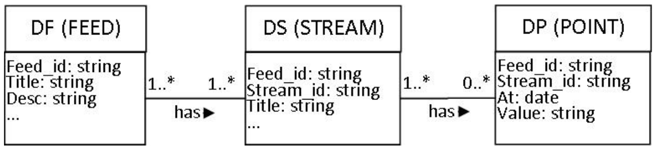

Fig. 1. Data hierarchy

The CPPM-TBAC algorithm is demonstrated in Fig. 2. First of all, two top-level visibility controls for resources are defined:

1. *Public access*: These resources can be searched and viewed by everyone and do not require a token.
2. *Private access*: These resources can only be accessed if a token with the necessary permissions is used. Child resources of a private visibility resource are always private.

Then the actions on these resources are categorised as either:

1. *Read actions*: Identified by the GET HTTP verb, these actions view resource information.
2. *Modify/write actions*: Any action that uses the remaining HTTP verbs has the potential to modify resources on the network. Regardless of the visibility of a resource, a token with the necessary permissions is required to carry out these actions.

CPPM defines two upper-level scopes when forming the tokens: (1) *Global scope* and (2) *Local scope*. The global scope can contain the basic grants (CRUD operations, i.e. create, read, update and delete) and the extended access restrictions. On the other hand, the local scope can only specify the basic grants for individual resources or a group of resources. Permissions defined in the global scope cascade to all public and private resources of the resource owner. The local scope can then be used to refine (extend/restrict) these permissions further if needed, or to remove certain resources from the permission set altogether.

The eventual applied access grants are calculated according to the following methodology:

1. If global grants are present and local grants are absent then apply the global grants on all public and private resources for the resource owner.
2. If local grants are present and global grants are absent then apply the local grants on the specified resources for the resource owner.
3. If both global and local grants are present, then do the following:
   a. Apply the global grants on all public and private resources of the resource owner;
   b. For the feeds and streams specified in local grants:
      i. Keep the global grants which have not been specified in the local scope.
      ii. Apply the local grants which have not been specified in the global scope.
      iii. Overwrite the global grants which exist in the local scope with the local scope grants.

This methodology is only applied on the basic grants and not on the extended access restrictions which are always defined in the global scope and cannot be overwritten locally.

In the global scope, the basic grants consist of the CRUD operations and any or all of these can be defined with a value of 1 (grant) or 0 (restrict). CPPM employs the *least access* methodology so that the absence of a grant is equal to its restriction.

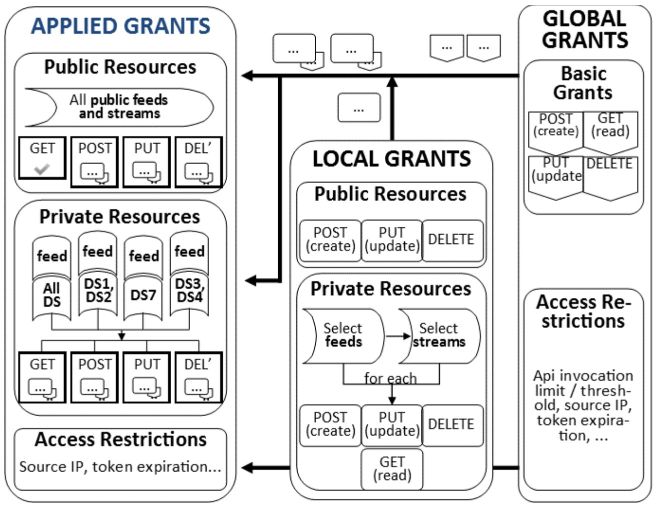

Fig. 2. Pictorial illustration of the CPPM Algorithm

A. Serial Sensor Payload Submission Procedure

The serial sensor payload submission procedure is shown in Fig. 3. It shows multiple devices being connected to a client, each sending sensor readings either periodically or when stimulated. The purpose of the client is to construct payloads for each device interaction. The payloads are constructed in a way such that they can be processed by the SAW network (if they are being submitted to the server) or the connected devices (if they are being submitted to the devices). Multiple devices can connect to the client at the same time.

The constructed payloads depend on the type of interaction. They can be one of the following:

1. A datafeed payload: This occurs when a new device wants to register with the SAW network or an existing device wants to update its definition. For example, this can happen if a user wants to register a new Arduino multi-sensor platform with the SAW network;
2. A datastream payload: This occurs when a datafeed wants to register a new datastream with the SAW network or wants to update an existing datastream belonging to it. For example, this can happen if a user had added a new sensor to his/her multi-sensor platform and wants to register the new sensor with the SAW network;
3. A datapoint payload: This occurs when a datastream wants to upload sensor readings to the SAW network. An example of this is a sensing device sending periodic readings to the SAW network.

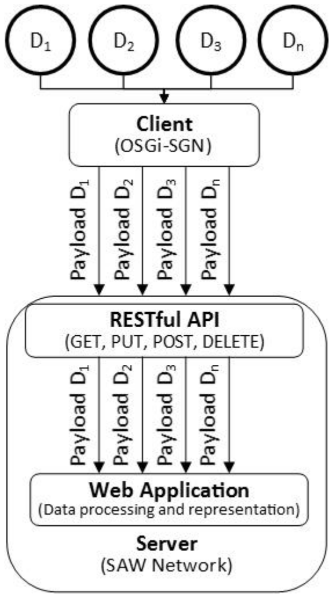

Fig. 3. TBAC serial payload submission procedure.

In the previously defined serial payload submission procedure [5], each payload is processed and transmitted to the SAW API sequentially by the client. For example, the client will submit the payload D1 to the SAW API, and then wait for a response. When it has received a response, it will send the next payload.

Consequently, the API receives and processes each payload in isolation of the other payloads. This means that the server needs to initialise a new processing action and a database connection for each payload it receives under this methodology. So for example, if *n* number of payloads are submitted in this manner and assuming that each payload uses the same access token, instead of the server having to check the access token only once, it will have to check it *n* times because each payload is captured and processed in isolation.

IV. PROPOSED EXTENION TO SAW

A. *Aggregated Sensor Payload Submission Procedure*

The proposed aggregated sensor payload submission procedure is shown in Fig. 4. It shows multiple devices being connected to a client, each sending sensor readings either periodically or when stimulated.

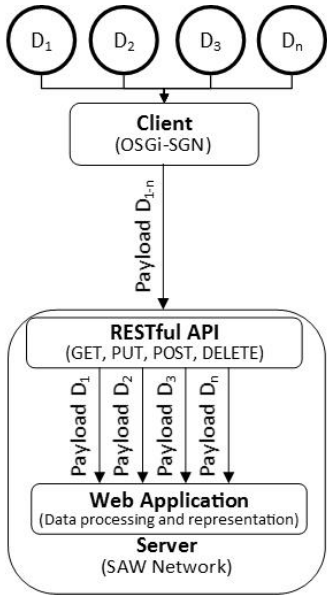

Fig. 4. TBAC aggregated payload submission procedure.

This procedure is quite similar to the previous procedure but varies in two major aspects:

1. At the client end: The client has to decide how many payloads to combine and how to package this combination as a new aggregated payload. It should be kept in mind that the current iteration of SAW only allows usage of a single access token for each request (whether it's a single payload or an aggregated payload). Thus, the client has to ensure that it only aggregates payloads for datafeeds, datastreams and datapoints that can be processed by the network with the supplied token. Since this intelligence is currently not available in the client node, for simulation purposes the payloads for aggregation are manually generated depending on the supplied token to ensure that the request is valid. For example, a payload is defined manually and then replicated the desired number of times whilst ensuring that all the generated payloads can be processed by the supplied token;
2. At the server API end: The server API has to be able to recognise an aggregated payload submission and then extract the individual payloads for processing. As mentioned in the previous point, the server expects a single access token with each request. This access token is used to check the associated grants stored in the database to determine whether the client's request can be fulfilled.

At the client end, one of the crucial decisions is determining the optimum number of payloads to combine in order to achieve the best possible performance metrics. This optimisation is not considered in this paper due to limitation in time and scope. Instead, payloads are aggregated on the fly for 100-1,000 devices and the results compared against the same payloads but submitted in the serial fashion.

In the current iteration, the payload aggregation creation procedure is pretty simple. First of all, an aggregated payload structure is created. This starts off as a blank payload. Then, each of the generated payloads is taken and appended to the aggregated payload. The final result is a well-constructed payload packaged in a representation format like JavaScript Object Notation (JSON).

B. *Payload Processing Procedure*

The payload processing procedure undertaken inside the server web application is shown in Fig. 5. The requests first pass through the RESTful API, and are then processed by the web application. The processed data is stored in the database for future interactions, and a response is sent back to the client. The response indicates the result of the payload submission request and includes any additional parameters required as part of the response (e.g. new device URI in the case of device registration).

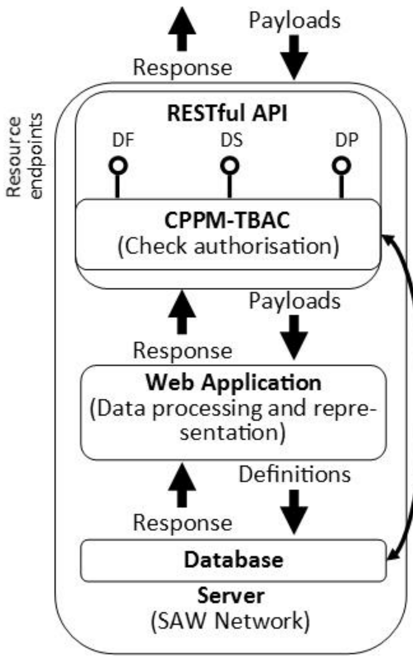

Fig. 5. SAW payload processing procedure.

The RESTful API consists of two major components:

1. The resource endpoints: These are specially designated URIs where resource interaction requests are handled with the use of the HTTP verbs. An example of a HTTP POST request to a resource endpoint for a datastream called “lightSensor”, belonging to a datafeed called “Arduino”, is as follows (this will update the “lightSensor” datastream in accordance with the provided payload):

```
POST
/api/v1/feeds/Arduino/streams/lightSens
or
```

2. The CPPM-TBAC: After a request comes into one of the resource endpoints, the CPPM-TBAC mechanism communicates with the database to authorise the request with the provided token. If the provided token has the necessary grants, the request is allowed to proceed ahead. Otherwise (or if no token is provided), the request is terminated and the user notified of having insufficient grants to carry out the associated request.

In should be noted that the CPPM-TBAC phase will not occur for publicly-exposed resources, since these are not protected and a token is not required to interact with them.

In both the serial sensor payload submission and the newly proposed aggregated sensor payload submission procedures, the CPPM-TBAC needs to communicate with the database to retrieve the access grants for the given token. Such database operations are quite costly, and needs to be repeated significantly more times in the serial sensor payload submission procedure because each payload is submitted in isolation of other payloads, and thus requires its own isolated processing.

However, with the aggregated sensor payload submission procedure, multiple payloads are received by the server at the same time. This allows the server to construct not only more optimised database queries but also reduce the number of database queries needed significantly by retrieving more data in each query. This results in less initialisations of database connections (typically just one), and as results indicate in the following section, dramatically reduces the round-trip response time of the payload submission requests.

V. Comparison of Serial and Aggregated Sensor Payload Submission Procedures

The simulation setup consists of an Open Service Gateway initiative (OSGi) Sensor Gateway Node (SGN) node acting as the client (and henceforth referred to as the *client*) and the SAW network acting as the server. The OSGi standard is a service-oriented component model which enables high modularity and portability of the codebase and improves resource utilization [11]. The SAW framework uses a combination of MySQL database for user management and logging and monitoring, and MongoDB (a No-SQL database) for storing tokens and sensing devices definition and data.

The tests are carried out for the new device registration interaction (submission of a new datafeed), with the number of devices ranging from 100 devices to 1,000 devices. The round-trip response times are measured both with CPPM-TBAC turned off and on.

Two important performance metrics are being measured in this comparison:

- The round-trip response time between the client submitting the request and getting a response from the server;
- The percentage delay added when CPPM-TBAC is turned on. The percentage delay added parameter was used in the preceding study to evaluate the scalability of the CPPM-TBAC scheme [5]. In this study, the focus is on comparing the difference between the two payload submission procedures and identifying any key trends.

A. Registering Datafeeds via Serial Payload Submission Procedure

The response times for registering 100-1,000 datafeeds using the serial sensor payload submission procedure are shown in Fig. 6 (with TBAC disabled) and Fig. 7 (with TBAC enabled).

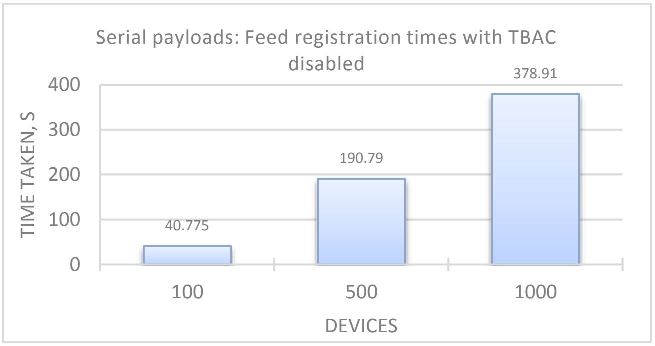

Fig. 6. Response times for registering devices using the serial sensor payload submission procedure with TBAC off.

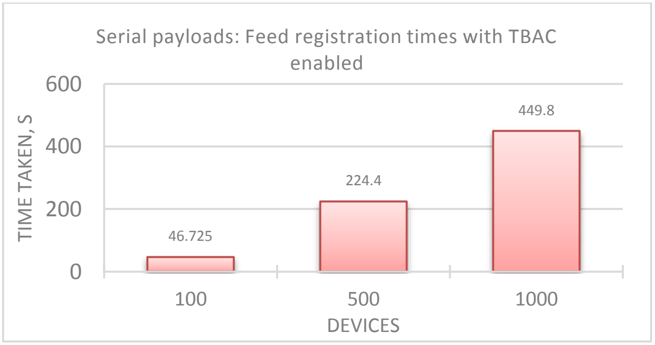

Fig. 7. Response times for registering devices using the serial sensor payload submission procedure with TBAC on.

Registration of 100 devices takes around 40 seconds when TBAC is disabled. This is increased to 46 seconds when TBAC is enabled, resulting in an increased delay of 14.6%. On the higher scale when registering 1,000 devices, it takes nearly 6 minutes and 19 seconds with TBAC disabled and 7 minutes and 30 seconds with TBAC enabled. This translates to an increased delay of 18.7% which is only marginally higher than the increased delay for 100 devices. The full set of comparisons are available in Table I and the added delay percentage plot can be seen in Fig. 8.

TABLE I. COMPARISON OF DEVICE REGISTRATION TIMES USING THE SERIAL SENSOR PAYLOAD SUBMISSION PROCEDURE WITH TBAC ON AND OFF.

	With TBAC disabled	**With TBAC enabled**
Registration of 100 devices	40.8 seconds	46.7 seconds (**14.6% slower**)
Registration of 500 devices	190.8 seconds	224.4 seconds (**17.6% slower**)
Registration of 1,000 devices	378.9 seconds	449.8 seconds (**18.7% slower**)

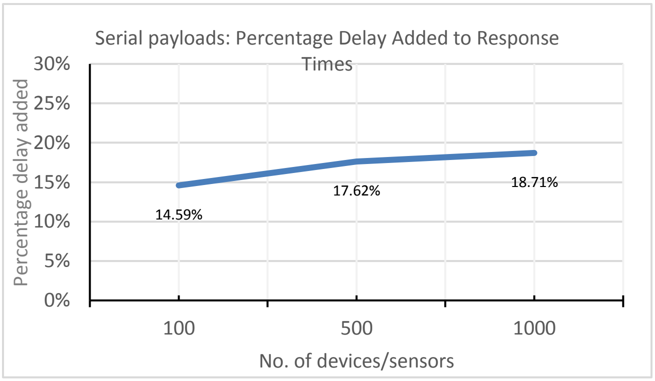

Fig. 8. Added delay percentage variation for device registrations using the serial sensor payload submission procedure.

B. Registering Datafeeds via Aggregated Payload Submission Procedure

The response times for registering 100-1,000 datafeeds using the aggregated sensor payload submission procedure are shown in Fig. 9 (with TBAC disabled) and Fig. 10 (with TBAC enabled).

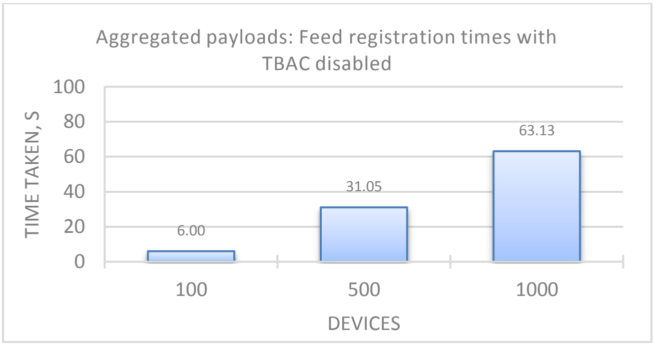

Fig. 9. Response times for registering devices using aggregated sensor payload submission procedure with TBAC off.

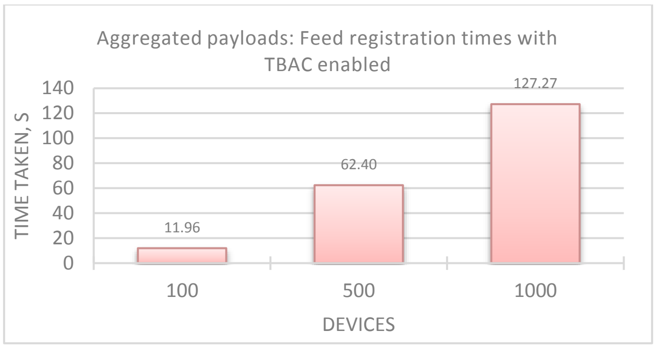

Fig. 10. Response times for registering devices using aggregated sensor payload submission procedure with TBAC on.

The full set of comparisons are available in Table II and the added delay percentage plot can be seen in Fig. 11.

TABLE II. Comparison of device registration times using the aggregated sensor payload submission procedure with TBAC on and off.

	With TBAC disabled	**With TBAC enabled**
Registration of 100 devices	6 seconds	11.958 seconds (**99.2% slower**)
Registration of 500 devices	31 seconds	62.4 seconds (**100.9% slower**)
Registration of 1,000 devices	63.1 seconds	127.3 seconds (**101.6% slower**)

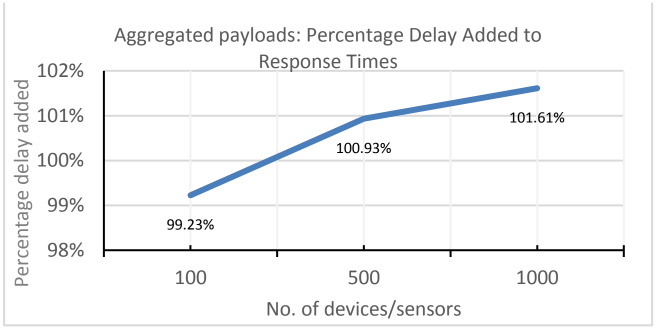

Fig. 11. Added delay percentage variation for device registrations using the aggregated sensor payload submission procedure.

Two things can be noted with these results instantly:

1. The response times are exponentially better in this scenario. The response times have improved by almost 700% when TBAC is disabled (Fig. 12) and nearly 400% when TBAC is enabled (Fig. 13);
2. The delay when TBAC is enabled is almost double compared to the serial sensor payload submission procedure.

In regards to the first point, it can be see here that aggregating payloads to reduce the number of requests made to the server greatly improves the response time. This is mainly due to the reduction in the number of database initialisations that need to be done, as this is the most costly operation on the server. Reducing the number of database initialisations leads to a great improvement in response times because the server can do more work with each database connection.

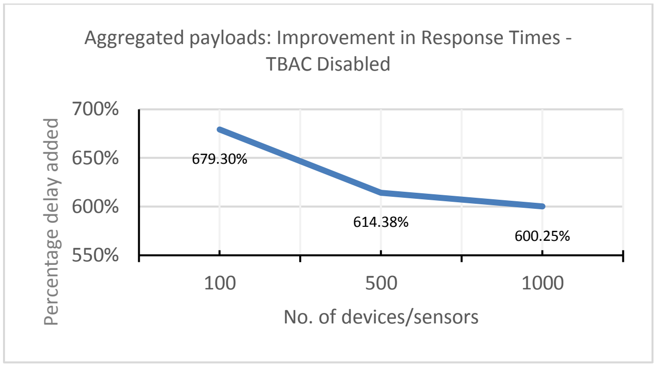

Fig. 12. Improvement in response times with TBAC disabled for aggregated payloads.

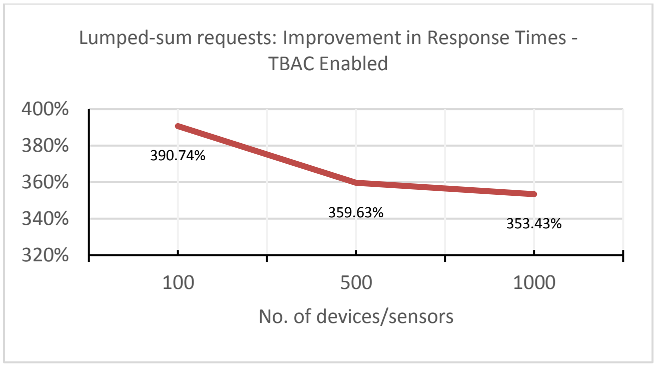

Fig. 13. Improvement in response times with TBAC enabled for aggregated payloads.

In regards to the second point, it can be seen in Table II that in this scenario, the response times double when TBAC is enabled. In comparison, the added delay in response times seen in the serial payloads scenarios was in the region of 15-30%. However, the increase of response times to just over 100% when TBAC is enabled in the aggregated payloads submission scenario can be easily explained.

When TBAC is enabled, the number of queries to the database increase significantly due to checking of permission policies for the supplied token. However, the added delay due to this process is relatively small compared to the time taken to initialise and close down the database, and is thus quite largely masked in the overall response time for serial payloads scenarios. For the aggregated payloads scenarios, however, this delay is more noticeable because the database is not being initialised or closed down again and again as the payloads are being processed. So in the aggregated payloads scenario, the actual added delay for using TBAC is being observed.

More importantly, it should be noted that once again, the added delay variation remains relatively uniform as the number of devices being registered are increased from 100 devices to 1,000 devices (Fig. 11). The added delay only increases by a mere 2.58% as the number of devices increases by 10 times from 100 devices, proving the CPPM-TBAC can scale efficiently with increasing number of devices in the WoT environment regardless of whether the payloads are submitted in a serial or an aggregated manner.

VI. Concluding Remarks

This paper has proposed a new aggregated CPPM-TBAC model for submitting sensor payloads to the server. The new model extends the previously defined serial sensor payload submission procedure by adding support for payload aggregation through OSGi-enabled sensor gateway nodes. The paper has also compared the previously defined serial and the newly proposed aggregated sensor payload submission models and procedures for capturing and submitting sensor data in the WoT. The methodologies for both procedures have been clearly demonstrated to identify the different characteristics of each technique.

It has been shown that the aggregated sensor payload submission procedure fares significantly better than the serial sensor payload submission procedure. In fact, an improvement of over 700% can be seen in the reduction of the round-trip response time when comparing the aggregated sensor payload submission procedure against the serial method. This is highly beneficial for improving the overall response time in the WoT.

Future work in this area can look at the effect of varying payload sizes for the submission procedures and analysing if this affects the response times. Another area of further exploitation can be the variation of the number of payloads that are aggregated and analysing the kind of effect this has the response times.

It is also evident that this study has not tracked the performance of the Central Processing Unit (CPU) while carrying out the simulations. A future extension of this work can look at the effects of the aggregation density (number of payloads combined into a single aggregated payload) on the processing power and memory usage of the server to see if the decreased response times are in fact beneficial in the whole scheme of things, or if the impact on the processing power required and memory used offset the advantages gained in response times.

References

[1] K. Janowicz, A. Bröring, . C. Stasch, S. Schade, T. Everding and A. Llaves, “A RESTful proxy and data model for linked sensor data,” *International Journal of Digital Earth,* vol. 6, no. 3, pp. 233-254, 2013.

[2] A. Broring, P. Mau´e, . C. Malewski and K. Janowicz, “Semantic mediation on the Sensor Web,” in *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*, Munich, 2012.

[3] D. Guinard and V. Trifa, “Towards the Web of Things: Web Mashups for Embedded Devices,” in *2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009)*, Madrid, Spain, 2009.

[4] G. Iachello and G. D. Abowd , “A Token-based Access Control Mechanism for Automated Capture and Access Systems in Ubiquitous Computing,” Georgia Institute of Technology, Atlanta, GA, USA, 2005.

[5] M. Amir, P. Pillai and Y. Hu, “Cascading Permissions Policy Model for Token-Based Access Control in the Web of Things,” in *Future Internet of Things and Cloud (FiCloud) 2014*, Barcelona, 2014.

[6] D. Miorandi, S. Sicari, F. D. Pellegrini and I. Chlamtac, “Internet of things: Vision, applications and research challenges,” *Ad Hoc Networks,* vol. 10, no. 7, pp. 1497-1516, 2012.

[7] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess and D. Savio, “Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services,” *IEEE TRANSACTIONS ON SERVICES COMPUTING,* vol. 3, no. 3, pp. 223-235, 2010.

[8] “A Token-Based Access Control System for RDF Data in the Clouds,” in *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, Washington, DC, USA, 2010.

[9] M. Amir, Y. F. Hu, P. Pillai and Y. Cheng, “Interaction Models for Profiling Assets in an Extensible and Semantic WoT Framework,” in *Wireless Communication Systems (ISWCS 2013)*, Ilmeanu, Germany, 2013.

[10] M. Amir, P. Pillai and Y. Hu, “A Generic & Extensible Asset Model for a Semantic Collaboration Framework,” *International Journal of Advanced Computer Technology (IJACT),* vol. 3, no. 1, pp. 88-96, 2014.

[11] M. Kuna et al., “Android/OSGi-based Machine-to-Machine context-aware system,” in *IEEE 11th International Conference on Telecommunications (ConTEL)*, Graz, 2011.