SIP-MBA: A Secure IoT Platform with Brokerless and Micro-service Architecture

Lam Nguyen Tran Thanh[1], Nguyen Ngoc Phien*[2], The Anh Nguyen[3], Hong Khanh Vo[4],
Hoang Huong Luong[5], Tuan Dao Anh[6], Khoi Nguyen Huynh Tuan[7], Ha Xuan Son[8]
VNPT Information Technology Company, Ho Chi Minh city, Vietnam[1]
Center for Applied Information Technology, Ton Duc Thang University, Ho Chi Minh City, Viet Nam[2]
Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam[2]
Department of Computer Science, Faculty of Electrical Engineering and Computer Science,
VSB-Technical University of Ostrava, Ostrava, Czech Republic[2]
FPT University, Can Tho City, Viet Nam[3,4,5,6,7]
University of Insubria, Varese, Italy[8]

Abstract— **The Internet of Things is one of the most interesting technology trends today. Devices in the IoT network are often geared towards mobility and compact in size, thus having a rather weak hardware configuration. There are many light weight protocols, tailor-made suitable for limited processing power and low energy consumption, of which MQTT is the typical one. The current MQTT protocol supports three types of quality-of-service (QoS) and the user has to trade-off the security of the packet transmission by transmission rate, bandwidth and energy consumption. The MQTT protocol, however, does not support packet storage mechanisms which means that when the receiver is interrupted, the packet cannot be retrieved. In this paper, we present a broker-less SIP-MBA Platform, designed for micro-service and using gRPC protocol to transmit and receive messages. This design optimizes the transmission rate, power consumption and transmission bandwidth, while still meeting reliability when communicating. Besides, we implement users and things management mechanisms with the aim of improving security issues. Finally, we present the test results by implementing a collect data service via gRPC protocol and comparing it with streaming data by using the MQTT protocol.**

Keywords—Internet of Things (IoT); gRPC; Single Sign-On; brokerless; micro-service; MQTT; message queue; security

I. Introduction

In recent years, Internet of Thing (IoT) applications have grown and applied in most fields such as smart city, healthcare, supply chains, industry, agriculture, etc. According to estimates, by 2025, the whole world will have approximately 75.44 billion IoT connected devices [1]. Timothy et al. [2] claimed that the IoT system architecture consists of 5 layers in order from low to high: Things, Connect, Collect, Learn and Do.

Specifically, the Things layer is the class of actuators to control physical devices or sensors to collect environmental parameters. The Connect layer connects devices with applications and users. The Collect layer is responsible for aggregating the data returned by the Things layer devices. Finally, Learn layer is used to analyze data to give suggestions to Do layer in response to received data. Eventually, the Things and Connect layers are the two most important ones because these provide the input data for the upper layers.

The Things layer consists of devices in the IoT which have limitations in network connectivity, power, and processing capabilities [3]. Therefore, the question is how can we optimize the processing capacity and power consumption of the devices, while still have to meet some requirements of communication speed as well as the confidentiality of information on transmission lines. This problem is determined by the protocol itself within the Collect class.

For interfaces in constrained networks, MQTT and CoAP are proposed to use [4]. We found that MQTT protocol had faster Packet transmission and creation time twice as fast as CoAP protocol. Besides, for developers of low bandwidth and hardware configuration devices, MQTT is the most preferred protocol [5]. Comparison of energy consumption, Mart et al. [6] found that the MQTT protocol consumes less energy than CoAP. For the above reasons, in this paper, we evaluate the aspects of power consumption, hardware capacity capacity as well as security risks (security risks) present of The platforms use the MQTT protocol.

The MQTT protocol uses a publish / subscriber [7] architecture, with the MQTT broker at the center. The MQTT subcriber (client), connects to the broker and sends messages to topics. Brokers rely on topic for packet routing, meaning that subscribers who subscribe to a topic receive all messages sent to that topic. This means that when the MQTT broker crashes, the whole system will be affected.

The MQTT protocol has four levels of QoS, ranging from 0 to 2[1] and these QoS levels are related to the level of confidence in the transmission of the packet (QoS has the lowest confidence level and QoS-2 has the confidence level. highest reliability). According to the article [8], the ratio of packet loss and packet transmission rate of QoS-0 level is highest. However, according to the article [9], the energy consumption of the QoS-0 level is only about 50% of the QoS-2 level and the communication bandwidth of QoS-0 is also lower than that of QoS-2. The MQTT protocol runs on TCP [10] and each TCP connection can only make one request and one response. This means that in order to send a message at the QoS-2 level, each client in the system using

[1]https://mqtt.org/

MQTT must make four TCP connections [11]. This results in systems using MQTT consuming a lot of bandwidth and the device's hardware processing capabilities. In addition, by using a publish / subscribe architecture, in order for subscribers to be ready to receive incoming messages on topics, the subscriber must maintain a persistent connection to the MQTT broker. The time to keep connection is determined by the parameter keep alive. Article [12] has shown that the shorter the keep alive cycle, the higher the energy consumption. On the other hand, according to the article [13], MQTT was created for transmission purpose. Therefore, it does not provide message storage capability and does not guarantee the order of the message when it reaches the receiver. These are also two important issues that need to be considered as the weaknesses of the system using MQTT.

In terms of security, the current MQTT protocol only provides identity, authentication and authorization for the security mechanism [14] but is very simple. Therefore, there are a lot of security risks. Lundgren et al. [15] indicates that we can obtain data by subscribing to any topic of the MQTT broker that is public on the internet - a serious security risk. If the attacker subscribes to MQTT topics with the client ID, the victim will experience a denial of service status and all information sent to the victim is passed on to the attacker - a serious risk factor [16]. Regarding the Authentication mechanism, MQTT supports authentication by username and password pairs, but the authentication mechanism is optional and not encrypted. The MQTT client authenticates itself by sending the username and password plaintext in the CONNECT package. Attacker attacks are made easily by blocking packets [14]. Zaidi et al. [17] survey of Shodan, the world's first IoT search engine for Internet-connected devices, found that there were 67,000 MQTT servers on the public Internet with most of them not authentic. MQTT has support for Authorization mechanism to access specific topic based on access list (ACL). This access list must be predefined in the configuration file of the MQTT broker and we must restart the MQTT broker service if we want to apply the new access list configuration. This is inconvenient and difficult to scale especially for systems with billions of devices and these devices may only have the authority to take action for a specified period of time on specific topics. Access control and authorization are a major challenge [18].

In this paper, we summarize the weaknesses of platforms using MQTT protocol, such as bandwidth issue, hardware processing power, power consumption, no message storage mechanism, and security risks. Since then, the paper proposes the SIP-MBA Platform model to use brokerless architecture to limit the weaknesses of brokered architecture. Our SIP-MBA Platform is designed in micro-service, devices and services communicate directly with each other by peer-to-peer communication model for fast communication speed. In the system architecture, we also added message queues to store messages when communicating between services, helping to ensure that when a service fails, it is still possible to receive messages during recovery. The communication process of the components in our SIP-MBA Platform will take place proactively, without requiring a constant connection to reduce energy and bandwidth consumption. To do this, we use gRPC as the communication protocol between devices and services. This work presents a method of combining gRPC and Oauth to complement the Authentication and Authorization mechanism for SIP-MBA Platform. Besides, we introduces a management model of users and things and channels information to prevent denial-of-service attacks and impersonation of participating systems.

The rest of paper is organized as follows. In Section 2 we provide knowledge about the technology used in this work. Section 3 discusses related work. In Section 4, we introduce our proposed system Architecture and in Section 5 is software architecture. Section 6, we implement Collect data service - this is the most important service in our architecture. Section 7, we discuss our test results. In Section 8, we conclude the paper and discuss the potential future work.

II. Background

A. gRPC and http/2

gRPC[2] (general-purpose Remote Procedure Calls) is an open-source high performance framework of Remote Procedure Call Protocol (RPC) developed by Google. gRPC is built on http/2 protocol. Improvements in http/2 over previous versions allow for better and more efficient http performance connections. One of the most important features of http/2 is multiplexing, which sends and receives multiple packets in a single connection. The activity comparison between http/1.1 and http/2 is shown in Fig. 1:

gRPC has four types of communication as follows [19]:

- Unary: Similar to traditional client-server communication. The client sends a request to the server, waits for the server to process it, and then returns the results to the client.
- Server streaming: In this mode, the client sends a request to the server and then waits for the server to return a stream of data. The client read messages from that stream until no more messages are returned. The order of messages for each stream is guaranteed to be the same between client and server.
- Client streaming: Similar to Server streaming RPCs, in this type, the client is the side that sends the data stream to the server, the server read the stream and perform the necessary processing, and then return the data to the client.
- Bi-direction streaming: This is the type of method where the data is sent in a stream from both client and server directions, the data stream in both directions is independent of each other, and the client and server can process that streamed data independently. That means when the client sends a message to the server, the server can process it to perform a certain task (while still receiving other messages) and send the result back to the client (while the client is still sending another message).

B. Oauth Protocol and Single Sign-On

Oauth basically is an authentication mechanism that enables third-party applications to be authorized by the user

[2]https://grpc.io/

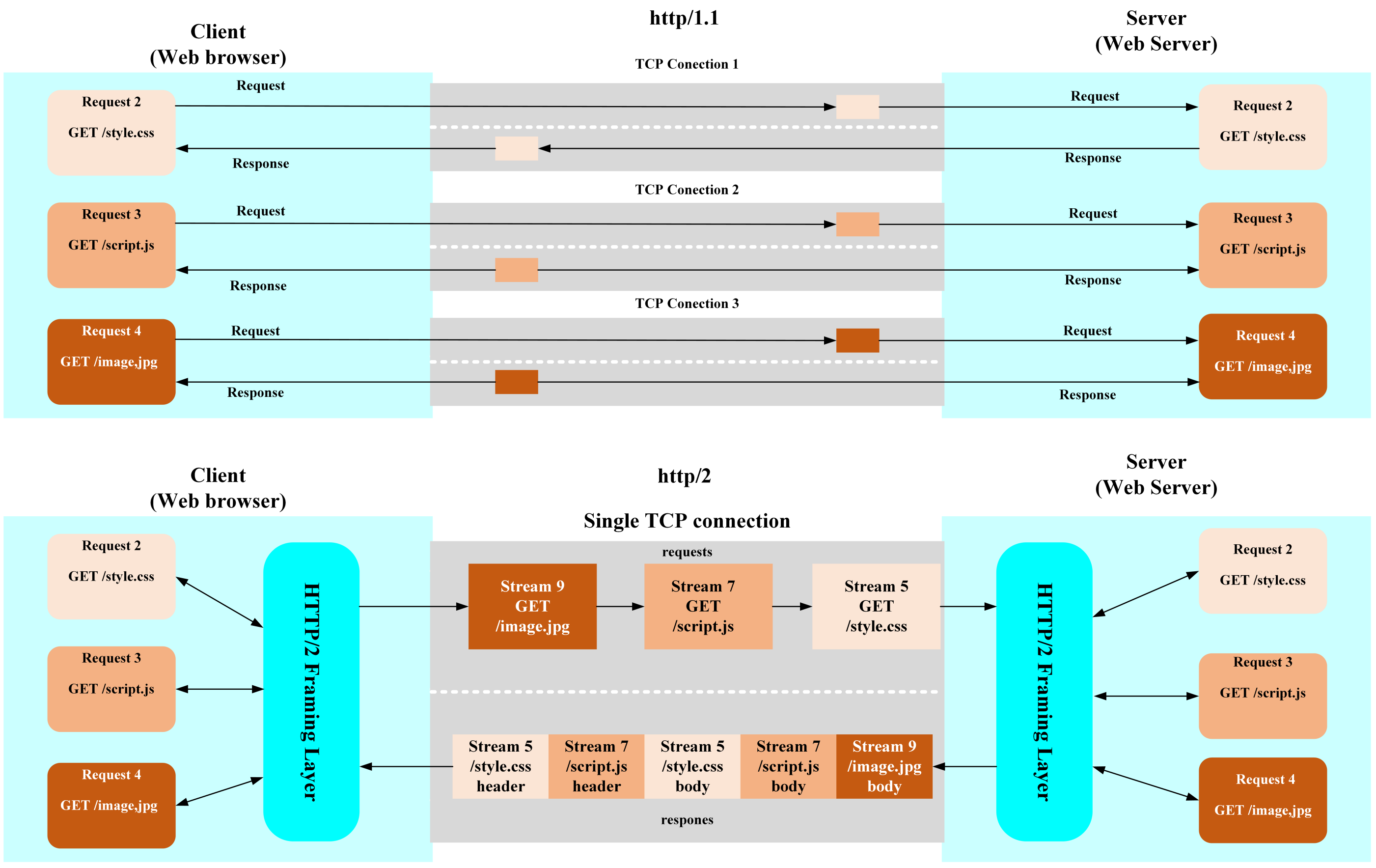

Fig. 1. The Activity Comparison between http/1.1 and http/2 [19].

to access the user's resources located on another application. Oauth version 2.0 is an upgrade of Oauth version 1.0, an authentication protocol that allows applications to share a portion of resources with each other without the need to authenticate via username and password as the traditional way. That helps limit the hassle of having to enter a username, password in too many places or register too many accounts for many applications that the user cannot remember all.

In Oauth, there are four basic concepts [3]:

- Resource owners: are the users who have the ability to grant access, the owner of the resource that the application wants to get.
- Resource servers: are the places which store resources, capable of handling access requests to protected resources
- Clients: are third-party applications that want to access the shared resource of the owner (i.e. prior to access, the application needs to receive the user's Authorization).
- Authorization servers: are the authentications that check the information the user sent from there, grants access to the application by generating access tokens. Sometimes the same Authorization server is the resource server.

[3]https://oauth.net/2/

Token is a random code generated by the Authorization server when a request comes from the client. There are two types of tokens, namely the access token and the refresh token. The former is a piece of code used to authenticate access, allowing third-party applications to access user data. This token is sent by the client as a parameter in the request when it is necessary to access the resource in the Resource server. The access token has a valid time (e.g., 30 minutes, 1 hour), when it expired, the client had to send a request to the Authorization server to get the new access token. Whereas, the latter is also generated by Authorization server at the same time with accessed token but with different function. Refresh token is used to get the new access token when it expires, so the validity period is longer than the access token.

Single Sign-On (SSO) is a mechanism that allows users to access multiple applications with just one authentication. SSO simplifies administration by managing user information on a single system instead of multiple separate authentication systems. It makes it easier to manage users when they join or leave an organization [20]. SSO supports many authentication methods such as Oauth, OpenID, SAML, and so on.

III. Related Work

A. *Brokerless Architecture*

Alif Akbar Pranata et al. [21] build a water quality monitoring system according to brokerless pub/sub architecture.

The paper uses two nodes including: relay node (publisher) to collect data from sensor and gateway node (subscriber) to receive data from relay node and send information to server. The paper uses Zigbee to send and receive messages in communication at close distances. However, the paper does not give methods to expand the system as well as security mechanisms.

Lam et al. [22] presented an architecture that combines MQTT broker and kafka message queue to connect different IoT service providers. This architecture allows individual service providers to communicate with each other easily without changing the existing architecture too much. In addition, Lam et al. [23] also evaluates power consumption, transfer speed, communication reliability, and security when using a combination of MQTT broker and kafka message queue. With Kafka's capabilities, we don't need to trade off transmission speed and reliability for power consumption (this is related to QoS-0 and QoS-2 levels). Moreover, these authors demonstrated an architecture that combines MQTT broker, Single Sign On, and kafka message queue [24]. This combination allows no need to trade-off speed and reliability when communicating with power consumption (this is related to QoS-0 and QoS-2 levels) while still ensuring security. of the system.

B. Oauth and Internet of Things

Paul Fremantle et al. [18] demonstrated using Oauth to enable to enable access control via the MQTT protocol. The results of the paper show that IoT client can fully use Oauth token to authenticate with MQTT broker. This indicates that Oauth is not only suitable for web application but also applicable for low hardware devices. The paper demonstrates implementing the Web Authorization Tool to create the access token, then embedded it (embedded) MQTT client. In addition, the paper presents the combined implementation of Oauth and MQTT for internal communication between MQTT broker and MQTT client. However, the paper uses RESTful over HTTP/1.1 to transfer the packet. This can be a waste of bandwidth and energy of IoT devices as discussed in Introduction. In our paper, we will use Oauth in conjunction with Single Sign-On to build into a service responsible for device authentication. Devices will communicate with this service through gRPC to take its advantages.

IV. System Architecture

The SIP-MBA Platform is designed in a micro-service architecture to ensure the system is horizontal scale that is shown in Figure 2. The architecture consists of three layers namely the Things layer, the Server layer and the User layer. Each class has components that play a different role.

The Things layer includes physical devices capable of collecting data from the environment (pH, temperature, humidity, etc.). After the collection process, the data are transferred to the Server layer according to the gRPC protocol. Things layer also receives a control command from the User layer or from the data processing service of the the Server layer to control the behavior of Things based on the information it collects.

The Server layer consists of micro-services that handle the following tasks:

- Collect data service: collect data from the (authenticated) things in Thing Layer. In addition, the collect data service also passes control commands from users or data processing service to things.
- SSO service: is responsible for authenticating things and users according to the Oauth protocol.
- Objects management service: allows to manage users and things participating in the system.
- Control service: allows the user to command control of things.
- Data processing service: analyze collected data to respond to users, store system logs and issue commands to control things according to pre-set triggers.
- Message Queue: is responsible for transporting messages interacting between services within the Server layer.

Finally, the User layer uses the Internet of Things service. Only users authenticated by the SSO service can interact with the Server layer.

V. Software architecture

To meet the goals set out by the SIP-MBA Platform, we provide several definitions of the components participating in the system and their interactions.

A. Users

Users are people who use IoT services. By constructing a user hierarchy model tree with the child's user_parent_id value equal to the parent user's username, our SIP-MBA Platform allows the creation and management of multiple levels of users without being bounded. The term depends on the characteristics of the organization. This tree-modeled user hierarchy makes the SIP-MBA Platform suitable for companies, especially when authorizing a specific user or changing the operation state for a series of child users at the same time w.r.t crashes (only ACTIVE users can request access token). User information is generated when a user registers to use an IoT service or is created by the parent user and provided to his or her child user.

Each user has a unique user_id value that conforms to the standard UUID[4] and is managed by the Objects-management service. When issuing commands to manipulate things, the user must pass in the access token obtained from the Single Sign On service. This token is authenticated by the Single-Sign-On service. In this way, we have enhanced the authentication mechanism, the authorization mechanism for the user as well as minimized the risk of a denial-of-service attack when a hacker impersonates a user repeatedly sending control commands.

B. Things

Things represent information about physical devices or applications owned by the user. To create things, the user that owns the things will have to pass in his valid Oauth token.

[4]https://tools.ietf.org/html/rfc4122

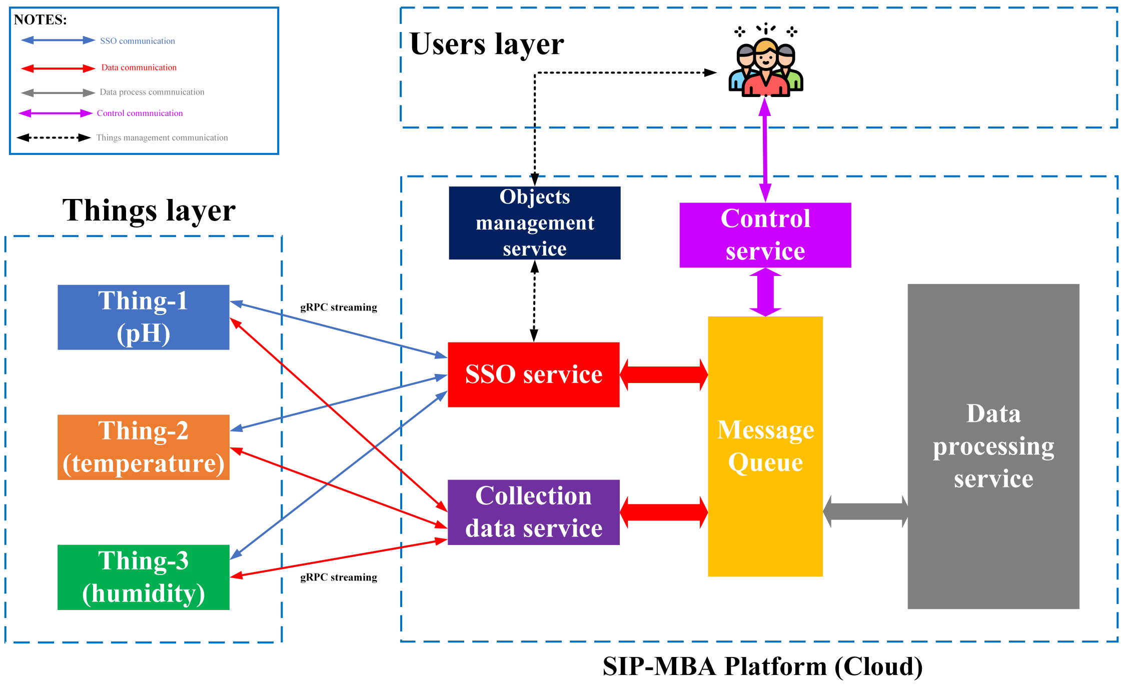

Fig. 2. SIP-MBA Platform as Micro-service Architecture.

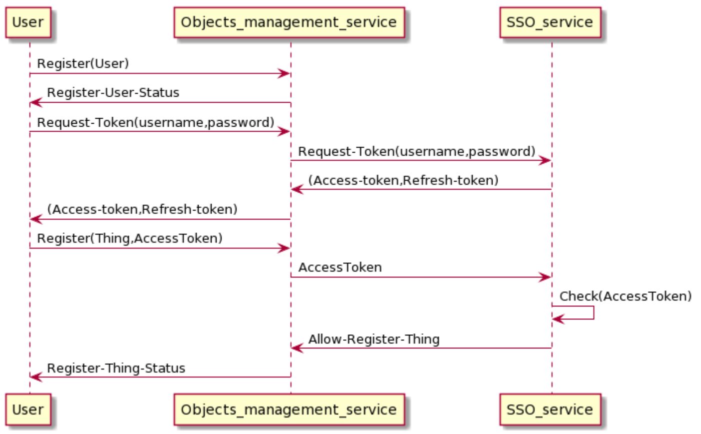
User
Objects_management_service
SSO_service
Register(User)
Register-User-Status
Request-Token(username,password)
Request-Token(username,password)
(Access-token,Refresh-token)
(Access-token,Refresh-token)
Register(Thing,AccessToken)
AccessToken
Check(AccessToken)
Allow-Register-Thing
Register-Thing-Status
User
Objects_management_service
SSO_service

Fig. 3. The User and Thing Initialization Processes.

After creating things, users must embed their access token into things. Embedding the access token is out of the scope of this paper. Only things that own a valid access token communicate with the Collect-data service. Similar to the user, this way to create device management information, helps increase security and reduce the risk of denial-of-service attacks.

C. Communication Workflow

1) Initialization workflow: As mentioned in Sections V-A and V-B, a user must be registered to use the Internet of thing service. After registration, the user makes a request for access token with username and password information. The system will return the user access token and refresh the token according to the Oauth standard. The user then performs registration of the thing by passing in his access token. The

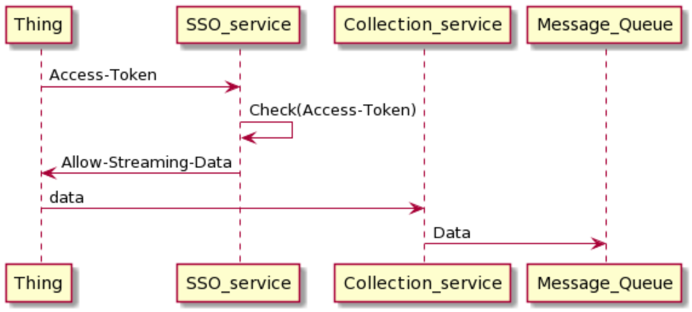

Fig. 4. Data Collection Process.

access token is authenticated by the SSO service. Finally, the Objects-management service will enable the creation of things information. The thread process initializes the user and thing as shown in Fig. 3

2) Data collection process: Things first send their access token to the SSO service. If the access token is authenticated by the SSO service then the thing is allowed to stream data to the Collection service. The collection service collects the data and passes it to the message queue. Other services like Data processing service access the data stream through the message queue. Thing after completing streaming data will disconnect from the Server layer and go to sleep to save energy. The protocol used in the Collect data process is gRPC. Process collect data is presented in Fig. 4

3) Control workflow: The user can issue commands to control the thing remotely by sending the access token and command to the Control service. The access token is authenticated through the SSO service. If the access token is valid, the

Control service forward the command to the Message Queue. Ultimately the collection service sends this command to things. The protocol used in the Control process is gRPC. The entire procedure is shown in Fig. 5

VI. Implementation

A. Database

In this paper we deploy the most important service is the collection data service, the remaining components will be present in our upcoming paper. The deployment model is shown in Fig. 6

The deployment model consists of two components. Firstly, the Raspberry Pi module plays the role of Thing in the System Architecture presented in Section 2. Raspberry Pi is a small computer packed with powerful hardware capable of running the operating system and installing many applications on it. With a price of only a few tens of dollars, Raspberry is currently the most prominent mini computer today. Initially, the Raspberry Pi Foundation developed the Raspberry project with the main goal of teaching computers to children and creating a cheap tool (only a few tens of dollars) for students to study and study. Currently, Raspberry Pi is commonly used in IoT systems because of its powerful processing capabilities, multitasking and exceptionally low cost. The second is Collect data service deployed on an Amazon EC2 server. The roles of the components are as follows:

- Raspberry Pi Module is responsible for collecting data from sensors and sending to the server via gRPC protocol.
- Collect data service is a service deployed on the server, in charge of receiving data sent from the client through the gRPC protocol.

The configuration of the two components is shown in Table I

VII. Evaluation

A. Environment Setting

After implementing the Collection data service, we set up a script to test the performance of the SIP-MBA Platform. This scenario tests the message delivery speed of a broker-less architecture, uses the gRPC protocol with a brokered architecture, and uses the MQTT protocol. The two of scenarios' source code is shown in our Github repository for collection data service[5] and MQTT streaming [6]. The testing scenarios are shown in Fig. 7.

We set up two test environments consisting of two Raspberry Pi modules and two Amazon EC2 Servers with similar configurations. The Raspberry Pi-1 and Server-1 pair implements the Collect data service and uses the gRPC protocol to transmit the packet. The Raspberry Pi-2 and Server-2 pair respectively deploy the MQTT client, the MQTT broker and use the MQTT protocol to transmit packets.

[5]https://github.com/thanhlam2110/iot-platform-collect-data-service
[6]https://github.com/thanhlam2110/mqtt-streaming

B. Message Delivery Speed Test Case

We perform a rate test of one million packets that are an integer between 1 and 1,000,000 on both test environments and record when the message was sent and when it was received. In our test, each sender message follows this structure **"hello + i"** with i comes from 1 to 1.000.000 For the environment using MQTT protocol, we perform the test with two levels of QoS-0 and QoS-2. The tests were performed 3 times and averaged. The results of the process obtained are shown in Table II

From the results noted in the Table II, we found that SIP-MBA Platform uses a broker-less architecture and gRPC protocol that has a much faster message delivery speed than systems using brokered architecture and MQTT protocol. Particularly in the case of MQTT QoS-2, Raspberry module can only send 65423, 65534 and 65672 messages, the MQTT broker hangs and does not continue to receive messages. This makes sense because gRPC runs on an http/2 protocol that allows sending and receiving messages with multiple packets in a single connection. Finally, gRPC is also a peer-to-peer communication instead of a broker like MQTT protocol.

C. Broken Connection Test Cases

In addition, we also have taken test scenario with broken connection between publisher and subscriber. We compare number of receive messages in case with and without using SIP-MBA platform when occurred broken connection. The test model is shown in the Fig. 8.

The test result show that, in case without using SIP-MBA platform, subscriber only receive one message - the newest message that publisher send when occurred broken connection. This is the retain function of MQTT protocol. When we enable retain flag, MQTT broker is ability to keep only newest message that publish by publisher. This message is received by subscriber after it reconnects to MQTT broker[9]. However, when using SIP-MBA platform, subcriber can receive all message that published by publisher. This is ability of kafka message queue therefore the system is guaranteed lost data.

D. Future Work

To develop a larger scenario and increase the number of devices/users authorized quickly, other security issues such as security, privacy, availability for objects are still the challenges. For the security aspect, further works will be deployed in different scenarios like healthcare environment [25], [26], [27], cash on delivery [28], [29]. For the privacy aspect, we will exploit attribute-based access control (ABAC) [30], [31] to manage the authorization process of the SIP-MBA Platform via the dynamic policy approach [32], [33], [34]. Finally, we will apply the blockchain benefit to improve the availability issues [35], [36], [37].

VIII. Conclusion

In this paper, we present the model of the SIP-MBA Platform in a broker-less and micro-service architecture. We use gRPC as the medium of communication to take advantage of its advantages. Besides, we also offer a solution to supplement

[9]http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

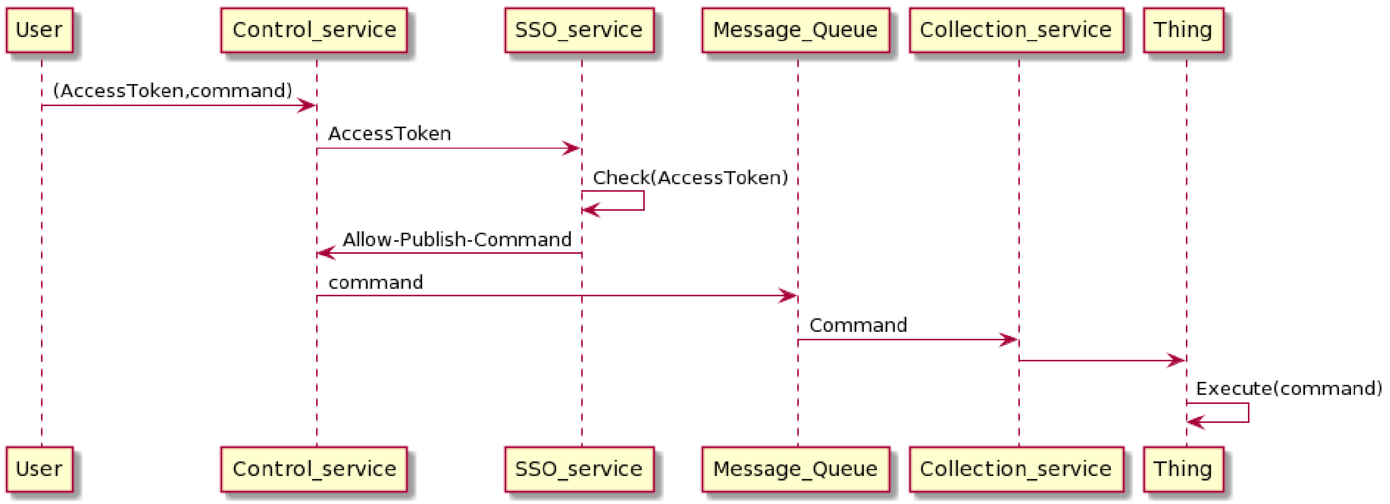

Fig. 5. Control Workflow.

TABLE I. Configuration of the Raspberry Pi Module and Collect data Service

	CPU	RAM
Raspberry Pi (module 3B)	Broadcom BCM2837, ARMv8 (64bit) quad-core, 1.2 Ghz	RAM 1GB
Collection Data Service (Server)	1vCPU	RAM 1GB

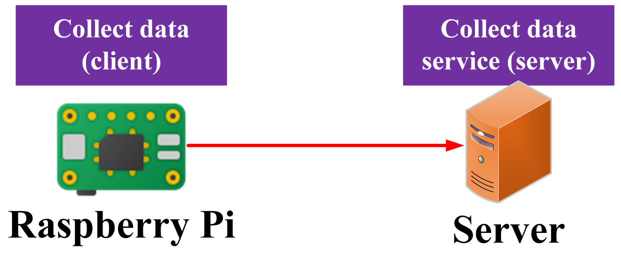

Fig. 6. The Collect Data Service Operators.

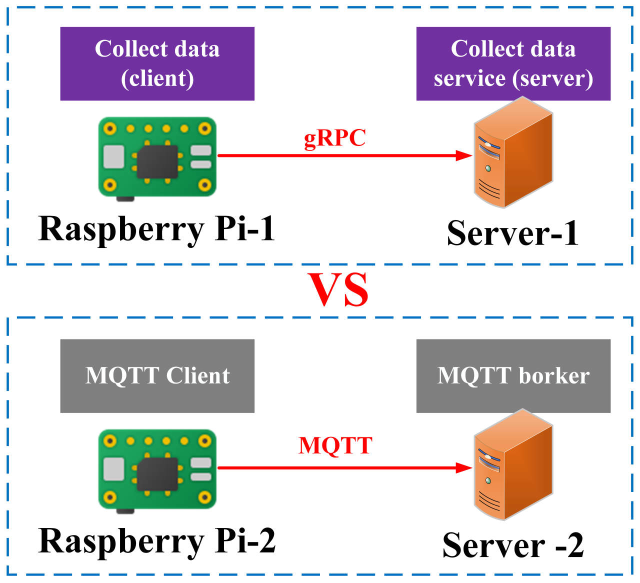

Fig. 7. The Scenarios of Communication Speed between gRPC and MQTT.

TABLE II. Results of Performance (seconds) Comparison between Two Scenarios (i.e., gRPC vs MQTT(QoS-0; QoS-2)).

Sending time	gRPC	MQTT (QoS-0)	MQTT (QoS-2)
1st	55s	266s	can't complete test
2nd	53s	268s	can't complete test
3rd	57s	272s	can't complete test
Average	55s	267s	can't complete test

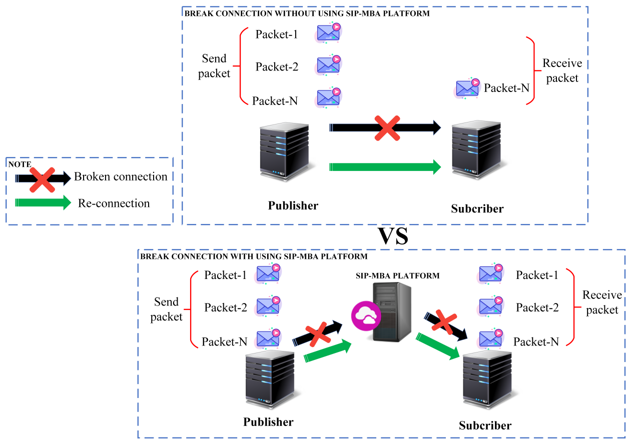

Fig. 8. Number of Receive Messages when System Recover after Broken Connection Issue.

the security mechanism for the system using Single Sign-On and Oauth. We also add a message queue system that stores and transports packets between services within the system. This ensures that when a service is not available, the message is still not lost. In the future, we will develop the remaining services and build a complete SIP-MBA Platform.

References

[1] T. Alam, “A reliable communication framework and its use in internet of things (iot),” *CSEIT1835111— Received*, vol. 10, pp. 450–456, 2018.

[2] T. Chou, *Precision-Principles, Practices and Solutions for the Internet of Things*. McGraw-Hill Education, 2017.

[3] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, “A survey on application layer protocols for the internet of things,” *Transaction on IoT and Cloud computing*, vol. 3, no. 1, pp. 11–17, 2015.

[4] S. P. Jaikar and K. R. Iyer, “A survey of messaging protocols for iot systems,” *International Journal of Advanced in Management, Technology and Engineering Sciences*, vol. 8, no. II, pp. 510–514, 2018.

[5] B. H. Çorak, F. Y. Okay, M. Güzel, Ş. Murt, and S. Ozdemir, “Comparative analysis of iot communication protocols,” in *2018 International symposium on networks, computers and communications (ISNCC)*. IEEE, 2018, pp. 1–6.

[6] M. Martí, C. Garcia-Rubio, and C. Campo, “Performance evaluation of coap and mqtt_sn in an iot environment,” in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 31, no. 1, 2019, p. 49.

[7] D. Soni and A. Makwana, “A survey on mqtt: a protocol of internet of things (iot),” in *International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017)*, vol. 20, 2017.

[8] S. Lee, H. Kim, D.-k. Hong, and H. Ju, “Correlation analysis of mqtt loss and delay according to qos level,” in *The International Conference on Information Networking 2013 (ICOIN)*. IEEE, 2013, pp. 714–717.

[9] J. Toldinas, B. Lozinskis, E. Baranauskas, and A. Dobrovolskis, “Mqtt quality of service versus energy consumption,” in *2019 23rd International Conference Electronics*. IEEE, 2019, pp. 1–4.

[10] B. Mishra, “Performance evaluation of mqtt broker servers,” in *International Conference on Computational Science and Its Applications*. Springer, 2018, pp. 599–609.

[11] N. Q. Uy and V. H. Nam, “A comparison of amqp and mqtt protocols for internet of things,” in *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*. IEEE, 2019, pp. 292–297.

[12] Y. Kwon and J. Lee, “Energy optimization model with variable keep-alive cycle algorithm in wireless sensor network,” *International Journal of Control, Automation and Systems*, vol. 17, no. 10, pp. 2531–2540, 2019.

[13] H. C. Hwang, J. Park, and J. G. Shon, “Design and implementation of a reliable message transmission system based on mqtt protocol in iot,” *Wireless Personal Communications*, vol. 91, no. 4, pp. 1765–1777, 2016.

[14] D. Mendez Mena, I. Papapanagiotou, and B. Yang, “Internet of things: Survey on security,” *Information Security Journal: A Global Perspective*, vol. 27, no. 3, pp. 162–182, 2018.

[15] L. Lundgren, “Light-weight protocol! serious equipment! critical implications!” *Defcon 24*, 2016.

[16] J. J. Anthraper and J. Kotak, “Security, privacy and forensic concern of mqtt protocol,” in *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM), Amity University Rajasthan, Jaipur-India*, 2019.

[17] N. Zaidi, H. Kaushik, D. Bablani, R. Bansal, and P. Kumar, “A study of exposure of iot devices in india: Using shodan search engine,” in *Information Systems Design and Intelligent Applications*. Springer, 2018, pp. 1044–1053.

[18] P. Fremantle, B. Aziz, J. Kopecký, and P. Scott, “Federated identity and access management for the internet of things,” in *2014 International Workshop on Secure Internet of Things*. IEEE, 2014, pp. 10–17.

[19] B. Pollard, *HTTP/2 in Action*. Manning, 2019.

[20] V. Radha and D. H. Reddy, “A survey on single sign-on techniques,” *Procedia Technology*, vol. 4, pp. 134–139, 2012.

[21] A. A. Pranata, J. M. Lee, and D. S. Kim, “Towards an iot-based water quality monitoring system with brokerless pub/sub architecture,” in *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2017, pp. 1–6.

[22] L. N. T. Thanh *et al.*, “Toward a unique iot network via single sign-on protocol and message queue,” in *International Conference on Computer Information Systems and Industrial Management*. Springer, 2021.

[23] ——, “Toward a security iot platform with high rate transmission and low energy consumption,” in *International Conference on Computational Science and its Applications*. Springer, 2021.

[24] ——, “Uip2sop: A unique iot network applying single sign-on and message queue protocol,” *IJACSA*, vol. 12, no. 6, 2021.

[25] H. X. Son and E. Chen, “Towards a fine-grained access control mechanism for privacy protection and policy conflict resolution,” *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 2, 2019.

[26] N. Duong-Trung, H. X. Son, H. T. Le, and T. T. Phan, “Smart care: Integrating blockchain technology into the design of patient-centered healthcare systems,” in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, ser. ICCSP 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 105–109.

[27] ——, “On components of a patient-centered healthcare system using smart contract,” in *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*. New York, NY, USA: Association for Computing Machinery, 2020, p. 31–35.

[28] H. T. Le, N. T. T. Le, N. N. Phien, and N. Duong-Trung, “Introducing multi shippers mechanism for decentralized cash on delivery system,” *money*, vol. 10, no. 6, 2019.

[29] N. T. T. Le, Q. N. Nguyen, N. N. Phien, N. Duong-Trung, T. T. Huynh, T. P. Nguyen, and H. X. Son, “Assuring non-fraudulent transactions in cash on delivery by introducing double smart contracts,” *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 5, pp. 677–684, 2019.

[30] N. M. Hoang and H. X. Son, “A dynamic solution for fine-grained policy conflict resolution,” in *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, 2019, pp. 116–120.

[31] H. X. Son and N. M. Hoang, “A novel attribute-based access control system for fine-grained privacy protection,” in *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, 2019, pp. 76–80.

[32] S. H. Xuan, L. K. Tran, T. K. Dang, and Y. N. Pham, “Rew-xac: an approach to rewriting request for elastic abac enforcement with dynamic policies,” in *2016 International Conference on Advanced Computing and Applications (ACOMP)*. IEEE, 2016, pp. 25–31.

[33] Q. N. T. Thi, T. K. Dang, H. L. Van, and H. X. Son, “Using json to specify privacy preserving-enabled attribute-based access control policies,” in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 2017, pp. 561–570.

[34] H. X. Son, T. K. Dang, and F. Massacci, “Rew-smt: a new approach for rewriting xacml request with dynamic big data security policies,” in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 2017, pp. 501–515.

[35] X. S. Ha, H. T. Le, N. Metoui, and N. Duong-Trung, “Dem-cod: Novel access-control-based cash on delivery mechanism for decentralized marketplace,” in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2020, pp. 71–78.

[36] X. S. Ha, T. H. Le, T. T. Phan, H. H. D. Nguyen, H. K. Vo, and N. Duong-Trung, “Scrutinizing trust and transparency in cash on delivery systems,” in *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 2020, pp. 214–227.

[37] H. X. Son, T. H. Le, N. T. T. Quynh, H. N. D. Huy, N. Duong-Trung, and H. H. Luong, “Toward a blockchain-based technology in dealing with emergencies in patient-centered healthcare systems,” in *International Conference on Mobile, Secure, and Programmable Networking*. Springer, 2020, pp. 44–56.