IoHT-MBA: An Internet of Healthcare Things (IoHT) Platform based on Microservice and Brokerless 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*— Internet of Thing (IoT), currently, is one of the technology trends that are most interested. IoT can be divided into five main areas including: Health-care, Environmental, Smart city, Commercial and Industrial. The IoHT-MBA Platform is considered the backbone of every IoT architecture, so the optimal design of the IoHT-MBA Platform is essential issue, which should be carefully considered in the different aspects. Although, IoT is applied in multiple domains, however, there are still three main features that are challenge to improve: i) data collection, ii) users, devices management, and iii) remote device control. Today's medical IoT systems, often too focused on the big data or access control aspects of participants, but not focused on collecting data accurately, quickly, and efficiently; power redundancy and system expansion. This is very important for the medical sector - which always prioritizes the availability of data for therapeutic purposes over other aspects. In this paper, we introduce the IoHT Platform for Healthcare environment which is designed by microservice and brokerless architecture, focusing strongly on the three aforementioned characteristics. In addition, our IoHT Platform considers the five other issues including (1) the limited processing capacity of the devices, (2) energy saving for the device, (3) speed and accurate of the data collection, (4) security mechanisms and (5) scalability of the system. Also, in order for the IoHT Platform to be suitable for the field of health monitoring, we also add realtime alerts for the medical team. In the evaluation section, moreover, we describe the evaluation to prove the effectiveness of the proposed IoHT Platform (i.e. the proof-of-concept) in the performance, non-error, and non affected by geographical distance. Finally, a complete code solution is publicized on the authors' GitHub repository to engage further reproducibility and improvement.**

***Keywords*—Internet of Health Things (IoHT); microservice; brokerless; gRPC; kafka; single sign-on; RBAC**

I. Introduction

The Internet of Thing (IoT) applications, currently, are increasingly diverse including smart city, healthcare, supply chains, industry, agriculture, etc. It is estimated that the number of IoT-connected devices worldwide will increase to 43 billion by 2023, three times that of 2018 [1]. Investments in IoT technology are expected to grow at 13.6% per year until 2022 [1], of which, the rate of IoT adoption in the healthcare sector accounts for 20%, second only to smart city with the density of 29% [2]. However, the IoT apps still have three main evaluation issues, namely: response time (27%), energy consumption (18%) and realiability (14%) [2]. Iot Platform is an intermediary system that acts as a glue to attach devices to users. Therefore, the IoT platform development optimizes the three aforementioned issues is very vital since it is a platform with all the more advanced features. There are many architectures for the IoT Platform, the most popularity is build in the five-layer architecture from low to high, namely Things, Connect, Collect, Learn and Do [3]. We found that the the main evaluation factors of IoT correspond to the design requirements of the three classes of Things, Connect and Collect.

Things (the physical devices) collect data directly or perform an action based on a control command from the user. The device is designed to fulfill the mobility requirements, hence it has limited power, processing capacity and bandwidth [4].

Connect includes tailored protocols that are suitable for the hardware and networking capabilities of the Things layer. Pratim et al. [5] evaluated the pros and cons of 9 protocols being applied for IoHT including: MQTT, CoAP, HTTP, XMPP, AMQP, RESTful, Websocket, SMQTT and DDS. In which, DDS (Data Distribution Service) protocol satisfies two criteria of response time and reliability. DDS uses M2M communication and brokerless architecture suitable for medical devices. However, the paper also points out the weakness of DDS is its low security and lack of support for many types of programming languages, especially python. Beside, the paper has not evaluate an assessment of the energy consumption of DDS compared to the rest of the protocols.

Collect layer is the collection software to collect data generated by Things through the Connect class. Therefore, the Collect layer architecture is determined by the protocol used in the Connect. This paper considers the brokering architecture at Collection layer with the MQTT protocol used in Connect layer, as this protocol is commonly used in IoT frameworks (including IBM, Amazon and Microsoft) [6]. The MQTT protocol uses a publish / subscriber architecture [7], with the MQTT broker at the center. The MQTT subscriber (client),

connects to the broker and sends messages to topics. Brokers rely on topic to route the packet, meaning that subscribers who subscribe to a topic will receive all messages sent to that topic. This can easily lead to a single point failure [8] at the central broker location. Besides, MQTT is created for transmission purposes. Thus, MQTT broker does not provide message storage capabilities and does not guarantee the order of messages when it reaches the receiver [9]. Finally, it also considers the weaknesses of brokering architectures using MQTT.

From the analysis of evaluation criteria and the three most basic functional layers of all IoT platforms, this paper focus on giving a new platform built on brokerless and microservice architecture for the healthcare environment (a.k.a IoHT-MBA Platform). We build the brokerless architecture on the Collection class using the gRPC protocol on the Connection class. Similar to DDS, the gRPC protocol also uses M2M communication and a brokerless architecture. However, gRPC supports more programming languages, which is convenient for developers. Also, gRPC consumes lower CPU and RAM compared to other IoHT protocols such as MQTT, CoAP, XMPP. The brokerless architecture does not have a central broker to coordinate topic-based messages, but instead is a service that communicates directly with the device to collect data. Collected data will be transferred to storage and distributed to advanced processing services (the function equivalent of Learn and Do classes). Besides, our IoHT-MBA Platform is designed based on the microservice architecture, ensuring fault tolerance, scaling horizontally, and ensuring the availability and capacity of the system as introduced in [10]. In addition, to ensure the security of the system, we also build a management model of the components participating in the IoHT-MBA Platform such as users and things. The management model is built according to Role-base Access Control (RBAC) architecture combined with hierarchical management of users according to the model tree.

The rest of the paper is organized as follows. In Sections 2 and 3, we provide the knowledge of the technology used and the work involved, respectively. Section 4 introduces the IoHT-MBA Platform and we build a prototype system to verify the effectiveness in Section 5. Section 6, we discuss the evaluation outcomes. Finally, Section 7 concludes the summarize and discusses further work.

II. Background

A. gRPC and http/2

gRPC[1] (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 allows us to send and receive multiple packets in a single connection. gRPC supports a variety of programming languages and is fully compatible with embedded devices [11]. This protocol provides four communication types as follows:

- 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. Kafka Message Queue

Kafka[2] is a distributed messaging system. It is capable of transmitting a huge number of messages in realtime. In the case that the receiver has not received the message, this message is still stored on the message queue and the disk to achieve the safety transaction.

The Kafka architecture includes the four main components, namely producer, consumer, topic, and partition. Kafka producer is a client to publish messages to topics. Data is sent to the partition of the topic stored on the broker. Kafka consumers are clients that subscribe and receive messages from topic, consumers are identified by group names. Many consumers can subscribe to the same topic. Data is transmitted in Kafka via the corresponding topic, when it is necessary to transmit data for different applications, it is possible to create many different topics. Partition is where to store data on the specific topic where each might have one or more partitions. On each partition, the data is stored permanently and assigned an ID called offset. Besides, a set of Kafka server (a.k.a broker and the zookeeper) is a service to manage the brokers.

C. OAuth and Single Sign-On Protocol

Oauth is an authentication mechanism that helps third party applications be authorized by the user to access the user resource located on the other application. OAuth version 2.0, an authentication protocol, is an upgrade version of OAuth version 1.0 and allows the applications to share a portion of resources with each other without provides the username and password as the traditional approach. That limits the hassle of having to enter the username, password in too many places or

[1] https://grpc.io/

[2] https://kafka.apache.org/

register too many accounts or applications. In OAuth includes four concepts[3].

- **Resource owners**: are users who have the ability to grant access, the owner of the resource that the application wants to collect.
- **Resource server**: a place to store the resource (resource), capable of handling access requests to protected resources.
- **Clients**: are third-party applications that want to access the shared resource as of the resource owner.
- **Authorization server**: authenticates, checks the information the user sent from there, grants access to the application by generating access tokens. Sometimes the authorization server is the resource server as well.

The token is a random code generated by the Proxy server when a request comes from the client. There are second type tokens which are i) access tokens and ii) refresh tokens. An access token is a code used to confirm access, allowing third-party applications to access user data. This token is sent by the client as a parameter in the request when needing access to the resource server's resource. The access token has a certain valid time (e.g., 30 minutes, 1 hour). When this expires, the client must request the proxy server to get the new access token. Although the refresh token is also generated by the proxy server simultaneously as the access token but differs in functionality. The refresh token is used to get a new access token when it expires, hence the longer duration than the access token.

D. Zabbix

Zabbix[4] is an open source that support to solve system monitoring issues. Zabbix could monitor network parameters, server status and data in real time. Zabbix offers the ability to set alarm thresholds and send alerts to subscribers via email, SMS or telegram, etc.

III. Related Work

A. IoT Architecture for Healthcare

Maktoubian et al. [12] has built an architecture that allows data collection from medical devices. This architecture collects data based on a combination of the MQTT protocol and the Kafka Message Queue. In addition, the architecture also performs big data analysis using Apache Spark. We appreciate the use of Kafka as it allows for secure data transmission. However, the use of MQTT protocol and brokering architecture brings many weaknesses as we discussed in Introduction. In addition, MQTT has 3 levels of Quality-of-Service (QoS) from 0 to 2. Selecting these QoS levels is a trade-off between reliability of packet transmission (rate of loss on the transmission line), transmission rate and energy consumption. The QoS-0 level has the fastest transmission rate but has the lowest reliability [13], the opposite is the QoS-2 level. According to the article [14] the energy consumption of the QoS-0 level is only about 50% of the QoS-2 level. The report also did not specify the level of MQTT QoS that the system uses. In terms of security, the article mentioned security risks, but did not specify how to fix it.

Taher et al. [15] building an IoT-cloud system to collect and process medical data. The system by the authors has full features, however, operating based on the MQTT protocol, there are many weaknesses in the security mechanisms [16] and the brokering architecture [8] that may cause single point failure.

Partha Pratim Ray [17], builds medical data collection system based on websocket and HTTP. However, according to Bansal et al. [5], websocket requires more RAM and HTTP header is a big challenge for low hardware devices. Besides, security aspect has not been considered.

Ha Xuan Son et al. [18] has built an emergency system for the patient. The system launched by the authors applies Blockchain technology on Hyperledger fabric platform with a strong focus on access control. However, the authors have not describe the collecting data method from patients. Moreover, the authors did not mention a specific architecture, so they cannot prove the scalability of the system. Also, the paper did not provide any evaluation on the processing speed of the system.

B. Microservice and Brokerless Architecture for IoHT

Jita et al. [19] developed an in-home medical care system (IHMCS), a system designed in a highly available and scalable micro service architecture. The system also uses blockchain to enhance security. However, in the implementation part, the system uses Zetta IoT Platform - a platform that uses HTTP and RESTful protocols to perform the communication process with the device. These two protocols are not suitable for devices with low hardware [5].

Martino et al. [20] provided a review of the most common architectural solutions available (i.e. pros and cons) to shape an IoT system, ranging from already standardized architecture to commercial ones. In healthcare fields, some studies [21], [22], [23] claimed the field of public healthcare using a service oriented architecture modeled as a multi-agent and multi-type policy system. Besides, these papers emphasized the importance of microservice in healthcare but it has not been clarified as a major contribution.

Di Zeng el at. [24] has launched research on medical systems according to micro service and brokerless architecture. This is a fairly complete functional system that a health care system must have and ensure scalability. However, the authors only give a research model but have not deployed the system.

Lam et al. [25] 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. [26] 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

[3]https://oauth.net/2/
[4]https://www.zabbix.com/

an architecture that combines MQTT broker, Single Sign On, and kafka message queue [27]. 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.

IV. System architecture

The proposed platform is designed according to microservice and brokerless architecture including three layers, namely, Things layer, User layer and Platform layer as shown in Fig. 1. The Things layer (physical device) the devices implement two independent services: collection data service (client) and control service (client). The former is responsible for streaming data collected from the environment according to a predetermined cycle to the collection data server (server) at the IoHT-MBA Platform. This data stream is only performed when the client is authenticated and checked the things' role by the Single Sign On service. The latter receives control commands from the control service (server) through the message queue system.

The user class (user group) registered to use the IoHT service. The user has the right to control and monitor the device state through the Control service (server), which will be performed after going through the authentication phase and verifying phase (i.e. the user's role) by the Single Sign-On service. Besides, the users can manage device information (for instance create / delete / disable / active) as well as manage their child users (for instance register / disable / active) through the Object Management Service. The information for managing the thing and the user is stored in the database and the Single Sign-On service also relies on this information to authenticate and verify the permissions of the Things and Users.

IoHT platform (system load balancer) includes Message Queue Service (MQS), Data Processing Service (DPS). MQS is in charge of routing the control packet from the Users to the Things and the data collected from the Things layer. Also, it stores messages going through the IoHT-MBA Platform, ensuring that when a service fails, it can still receive messages after recovery. Whereas, DPS does data analysis in depth. DPS is in charge of analyzing data according to medical parameters predetermined by IoHT service provider and as the input of Zabbix server. The Zabbix server system allows the user to configure the warning thresholds (a.k.a. the trigger) that sends notification to the user via the telegram-bot when the data output of the data processing service hits the threshold. For example, the user can configure to send the notification when the patient's heart rate reaches over 140bmp in 5 minutes. This post focuses on the three layers of things, connect and collect so we are temporarily using Telegram as the third party to send the notification to users.

V. Implementation

The article implementing the services outlined in the proposal section includes: collection data service[5], single sign on service[6], control service[7], message queue service[8] and object management service[9]. The deployment model and interaction between the symbolic services in Fig. 1 is as follows: (1a-7a) collect the data handler and send the notification, (1b-5b) control the device flow, (1c) manage the object.

The collect data stream details are shown in Fig. 2. Things collects data from sensors and periodically sends them back to the IoHT-MBA Platform (sending data periodically to help save energy). The collection data service (client) on things creates a single gRPC connection that both sends token credentials, role checks, and streaming data. This is the advantage that gRPC brings, discussed in Section 2.1 (process 1a). The collection data service (server) passes the token and role information to the single sign on service (process 2a), the test result is returned (process 3a). If things is authenticated and has a valid role, it will start streaming bulk of collected data, whereas Collection data service (server) will close gRCP connection. Data received by the Collection data service (server) will be sent to the Kafka Message Queue (process 4a) and forwarded to an in-depth analysis at the Data Processing service (process 5a). Data after being analyzed later will be monitored by Zabbix server (process 6a). The user has also set the notification threshold on the Zabbix server. When data read by the Zabbix server hits the threshold, it sends an notification to the user via Telegram-bot (process 7a).

Details of flow control are shown in Fig. 3. User sends information about the `access token`, `thingID` and `control command` to the Control service server (process 1b). The control service server will pass the above information to the Single Sign-On server to authenticate the user through the token, check the users'role based on the control command submitted by the user and check if the user owns it. On the other hand it is assigned the thing based on thingid (process 2b). The test result is returned to the Control service server (process 3b), if it is valid this service sends the control command to the Message Queue (process 4b). The Control service (client) deployed on Things opens a gRPC connection to read all the control commands in a pre-set cycle (process 5b).

Object flow is simply we provide APIs for users to interact with the database. The Single Sign-On service will also use the data stored here to authenticate and verify the roles of users and things.

We apply RBAC model in conjunction with Single Sign-On to provide role-based authentication and authorization for Things and Users. However, the concepts of user roles and things are different. We define the users' role including permissions that affect the thing and child user such as: create or delete thing/assign or unassign thing (for another user) / control thing / active or disable child user. Whereas things' roles are related to their services; for instance things are capable of collecting 3 information such as: heart rate, blood pressure and coordinates. If the thing don't have role **"heart rate"**, it could not allowed to send this data to the IoHT-MBA Platform, although still collected and stored locally. While if it have role **"blood pressure"** and **"coordinates"**, it could send blood pressure and coordinates data to the IoHT-MBA Platform. These role of things only set by the owner user.

[5]https://github.com/thanhlam2110/collection-service-new
[6]https://github.com/thanhlam2110/service-sso
[7]https://github.com/thanhlam2110/user-control-service
[8]https://github.com/thanhlam2110/tcp-kafka-producer
[9]https://github.com/thanhlam2110/object-management-service

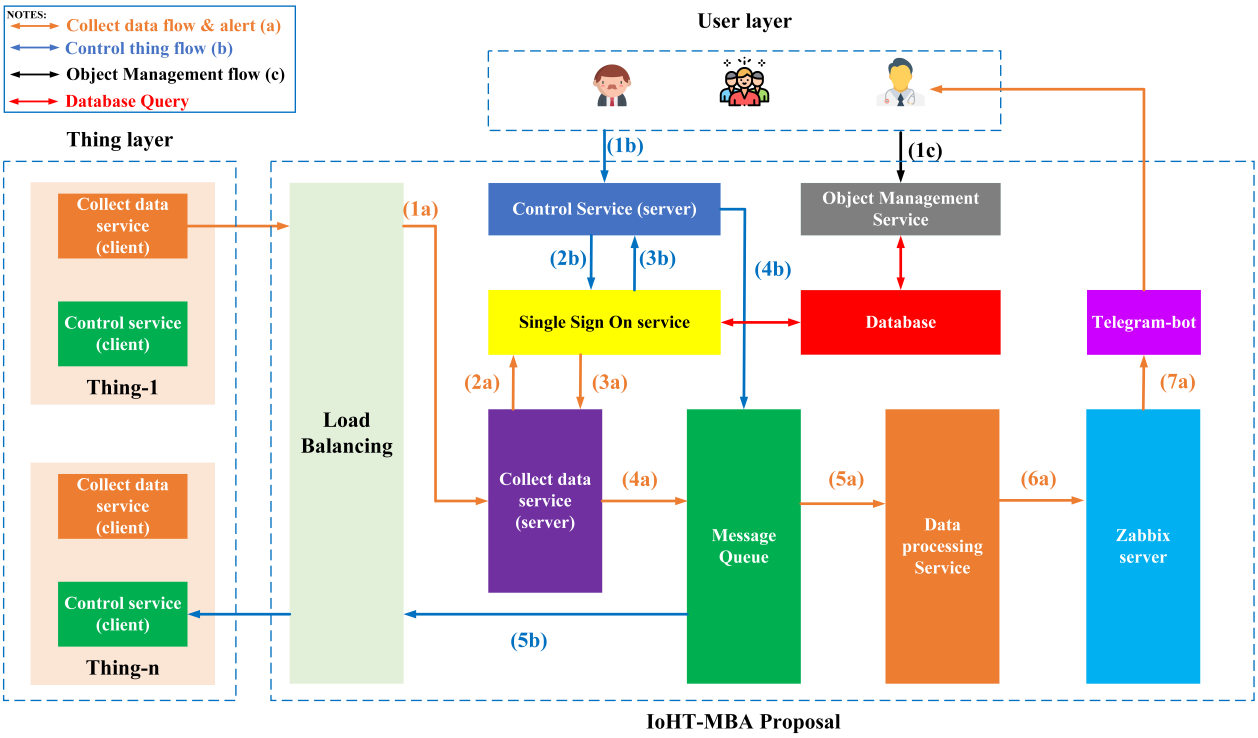

Fig. 1. IoHT-MBA Platform Proposal.

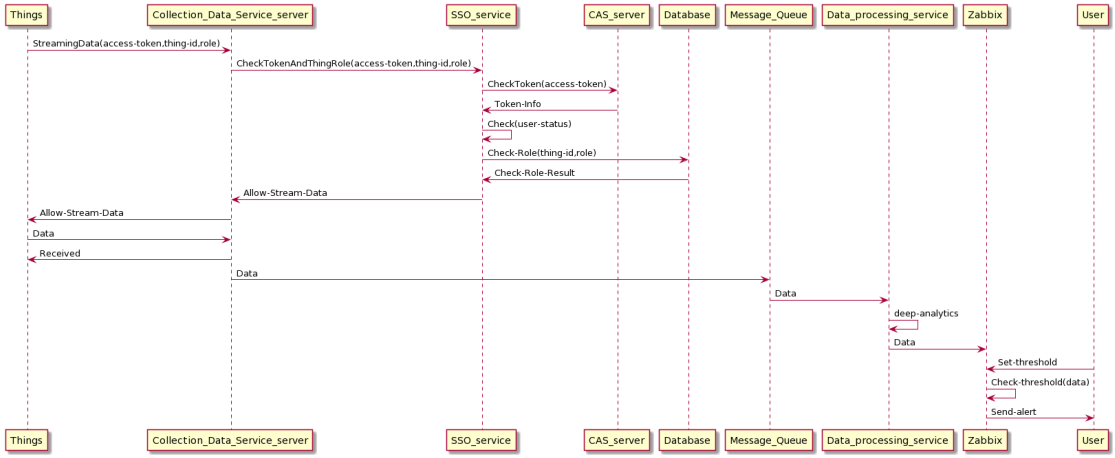

Fig. 2. Collection Data and Send Alert Flow.

This allows you to be flexible and proactive in sharing their information.

We implement the user management by model tree with the child user's `user_parent_id` field equal to the parent user's username. This allows our IoHT-MBA Platform to be flexibly applied to the business of performing hierarchical HR management. A parent user can `DISABLE` all of their child users (via user-status values) on the fly.

VI. Evaluation

A. Environment Setting

Our IoHT-MBA Platform is designed by microservice architecture. To perform the evaluation, we deploy these services to the Amazon EC2[11] platform with each service equivalent to a virtual machine with 1GB RAM and 1 vCPU configuration. For the device, we deploy collection data service (client) and control service (client) on the Raspberry Pi 3 model B+[12] module with Broadcom BCM2837, ARMv8 (64bit) quad-core, 1.2 GHz and 1GB RAM.

[11]https://aws.amazon.com/
[12]https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

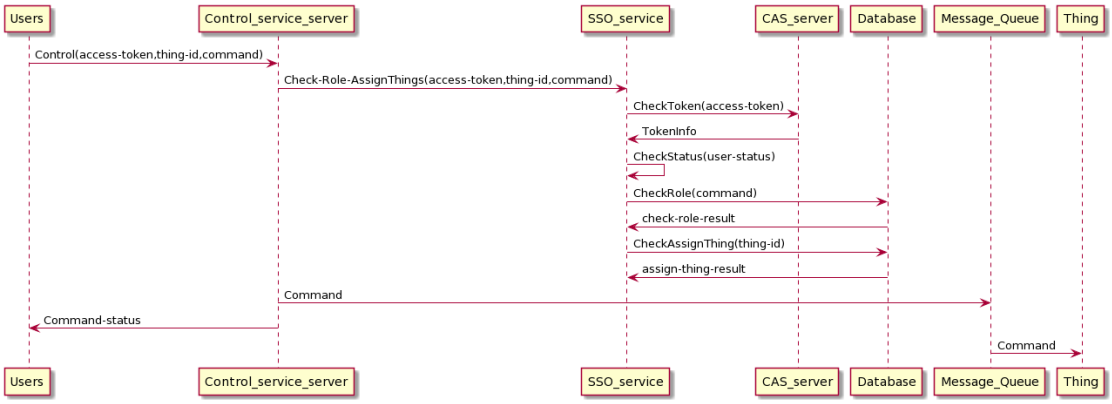

Fig. 3. The Control Flow.

B. Round Trip Time Test Cases

We measure Round Trip Time (RTT) from the moment things are streaming data to receiving it at Message Queue. We also look at the error rate (number of messages lost out of total messages). In addition, we also create EC2 VMs with a configuration equivalent to the Raspberry Pi model B + module in different geographical areas (except in Vietnam deployed on the real module) to simultaneously assess the impact of the site location. the delay time and error rate when streaming data Measured results are shown in Table I. The measurement results show that when we only deploy evaluation on low-profile servers, the processing time is completely unaffected by geographical distance. We realize that the data transfer rate is very fast and the error rate is 0, which is very suitable when applying the IoHT-MBA Platform for medical applications. All messages are received in full and in order. By using gRPC which allows sending multiple messages on one connection, we achieve very fast transfer speed while still satisfy the whole process of checking and validating before streaming data as described in Section 5.

C. Broken Connection Test Cases

Also, we have taken test scenario with broken connection between publisher and subscriber. We compare number of receive messages in case with and without using IoHT-MBA platform when occured broken connection. The test model is shown in the Fig. 4.

The test result show that, in case without using IoHT-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[13]. However, when using IoHT-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

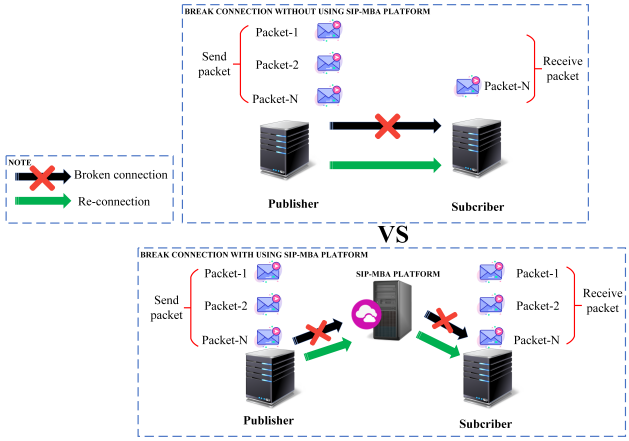

Fig. 4. Number of Receive Messages when System Recover after Broken Connection Issue.

security, privacy, availability for objects are still the challenges. For the security aspect, further works will be deployed in different scenarios like healthcare environment focusing on the smart care approaches [28], cash on delivery (CoD) [29], [30]. For the privacy aspect, we will exploit attribute-based access control (ABAC) [31], [32] to manage the authorization process of the IoT Platform via the dynamic policy approach [33], [34], [35]. Besides, we will apply the blockchain benefit to improve the availability issues [18], [36], [37]. Finally, we have a plant to develop data processing service to analyze the collected data.

VII. Conclusion

In this paper, we present the model IoHT-MBA Platform according to brokerless and microservice architecture. The scope of articles is oriented towards architecture that meets the most basic requirements and satisfies the top three evaluation criteria of IoHT Application, namely response time, energy consumption, and realibility. The microservice and brokerless architecture allows us to scale the system horizontally, while also ensuring availability, which is crucial in real systems. We take advantage of gRPC's advantages to define the data collection protocol. We also added the message queue system to store and transport packets between services within the

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

TABLE I. THE TEST RESULT

Location	Factor	1000	5000	10000	50000	100000
N.California	**RTT(s)**	3.30	14.02	26.17	132	261.38
	Error(%)	0	0	0	0	0
Stockholm	**RTT(s)**	3.27	14	26.05	131.55	261.5
	Error(%)	0	0	0	0	0
Ho Chi Minh city	**RTT(s)**	3.15	13.65	25.66	130.51	260.66
	Error(%)	0	0	0	0	0
Sydney	**RTT(s)**	3.19	13.72	25.8	131.05	261.15
	Error(%)	0	0	0	0	0

system. This ensures that when a service is not available, the message is still not lost and does not cause errors for the whole system. In addition, we also offer a solution that complements the security and decentralization of users and things through the RBAC model using Single Sign-On and OAuth. IoHT-MBA Platform is an open system, when we ensure a fast, accurate and energy-efficient data collection platform, developing advanced application layers using these data will be advantageous. more for developers. In the future, we will upgrade the platform's authentication and authorization mechanism by using Attribute-based Access Control to accommodate the increasingly complex nature of controlling things and users participating in the IoHT-MBA Platform.

REFERENCES

[1] F. Dahlqvist, M. Patel, A. Rajko, and J. Shulman, "Growing opportunities in the internet of things," *McKinsey, July*, 2019.

[2] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Internet of things applications: A systematic review," *Computer Networks*, vol. 148, pp. 241–261, 2019.

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

[4] A. Pratap, R. Gupta, V. S. S. Nadendla, and S. K. Das, "On maximizing task throughput in iot-enabled 5g networks under latency and bandwidth constraints," in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2019, pp. 217–224.

[5] M. Bansal *et al.*, "Application layer protocols for internet of healthcare things (iot)," in *2020 Fourth International Conference on Inventive Systems and Control (ICISC)*. IEEE, 2020, pp. 369–376.

[6] J. C. Fuentes Carranza and P. W. Fong, "Brokering policies and execution monitors for iot middleware," in *Proceedings of the 24th ACM Symposium on Access Control Models and Technologies*, 2019, pp. 49–60.

[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] P. Lv, L. Wang, H. Zhu, W. Deng, and L. Gu, "An iot-oriented privacy-preserving publish/subscribe model over blockchains," *IEEE Access*, vol. 7, pp. 41 309–41 314, 2019.

[9] 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.

[10] M. Ali, S. Ali, and A. Jilani, "Architecture for microservice based system. a report," 2020.

[11] N. Karcher, R. Gebauer, R. Bauknecht, R. Illichmann, and O. Sander, "Versatile configuration and control framework for real time data acquisition systems," *arXiv preprint arXiv:2011.00112*, 2020.

[12] J. Maktoubian and K. Ansari, "An iot architecture for preventive maintenance of medical devices in healthcare organizations," *Health and Technology*, vol. 9, no. 3, pp. 233–243, 2019.

[13] M. B. Yassein, M. Q. Shatnawi, S. Aljwarneh, and R. Al-Hatmi, "Internet of things: Survey and open issues of mqtt protocol," in *2017 international conference on engineering & MIS (ICEMIS)*. IEEE, 2017, pp. 1–6.

[14] 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.

[15] N. C. Taher, I. Mallat, N. Agoulmine, and N. El-Mawass, "An iot-cloud based solution for real-time and batch processing of big data: Application in healthcare," in *2019 3rd International Conference on Bio-engineering for Smart Technologies (BioSMART)*. IEEE, 2019, pp. 1–8.

[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] P. Pratim Ray, D. Dash, and N. Moustafa, "Streaming service provisioning in iot-based healthcare: An integrated edge-cloud perspective," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 11, p. e4109, 2020.

[18] 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.

[19] H. Jita and V. Pieterse, "A framework to apply the internet of things for medical care in a home environment," in *Proceedings of the 2018 International Conference on Cloud Computing and Internet of Things*, 2018, pp. 45–54.

[20] B. Di Martino, M. Rak, M. Ficco, A. Esposito, S. Maisto, and S. Nacchia, "A survey: Internet of things reference architectures, security and interoperability."

[21] R. Hill, D. Shadija, and M. Rezai, "Enabling community health care with microservices," in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*. IEEE, 2017, pp. 1444–1450.

[22] N. Duong-Trung, H. X. Son, H. T. Le, and T. T. Phan, "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.

[23] 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.

[24] D. Zheng, X. Zhang, and L. Chen, "Research of new integrated medical and health clouding system based on configurable microservice architecture," in *2020 IEEE 23rd International Conference on Computational Science and Engineering (CSE)*. IEEE, 2020, pp. 78–85.

[25] 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.

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

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.