Improvements of MQTT Retain Message Storage Mechanism

Xin Wu
School of Computer Science & Technology
Wuhan University of Technology
Wuhan, China
13628674612@163.com

Ning Li
School of Computer Science & Technology
Wuhan University of Technology
Wuhan, China

***Abstract*—Aiming to the disadvantage that MQTT agreement only saving one latest retain message which may result in missed and duplicate messages, it redesigned SUBSCRIBE and PUBLISH control packet format of MQTT, and realized new retain messages storage model in server and sending and receiving mechanism of retain messages. The experimental results showed that when publisher frequently publishes messages and subscriber disconnect, the subscriber will receive all the missed messages after being reconnected under new mechanism, while only the last missed message was received under old mechanism. When subscribers disconnect frequently and the server holds the retain message, subscribers received 0% repetitive messages under new mechanism while 100% repetitive messages was received under old mechanism. It proves that the new retain message storage mechanism can effectively prevent missing messages and repeatedly receiving messages.**

Keywords—MQTT(Message Queuing Telemetry Transport); Retain Message; Publish/Subscribe

I. INTRODUCTION

MQTT[1~4] is a publish/subscribe[5,6] based instant messaging protocol developed by IBM in 1999. The MQTT protocol specifies the mechanism for keeping server messages: if the retain flag for the PUBLISH packet sent by a client to a server is set to 1, the server must store this message and its QoS (Quality of Service)[7], so that it can be delivered to future subscribers whose subscriptions match its topic name[8,9]. However, this mechanism only saves the latest message under the topic. When the subscriber reconnects after disconnecting, all the messages except the latest one published during the disconnection will be missed or the latest message under the topic will be repeatedly received. Therefore, this paper made improvements to the MQTT retain message storage mechanism to solve the above problems.

II. MQTT PROTOCOL AND ANALYSIS

A. MQTT Control Packet

MQTT control packet[1] consists of up to three parts: fixed header, variable header and payload. Each MQTT control packet must contains a two-byte fixed header, while the last two parts may be not needed for some control packet.

A PUBLISH packet is sent from a publisher to a server or from server to a subscriber to transport an application message. A SUBSCRIBE packet is sent from a subscriber to a server to create one or more subscriptions, so that the server can send messages that match these subscriptions to the subscriber. Fixed header format is shown in Table 1, and more detailed can be seen in [1].

TABLE I. FIXED HEADER FORMAT

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Byte 1	MQTT Control Packet Type				Flags Specific to Each MQTT Control Packet Type			
Byte 2...	Remaining Length							

B. Retain Message Handling Mechanism

To save a message on the server for a long time, and every subscriber can receive the message when subscribed to the topic, MQTT provides the retain mechanism. The mechanism declared that if the retain flag is set to 1, in a PUBLISH packet sent by a publisher to a server, the server must store the application message and its QoS, so that it can be delivered to future subscribers whose subscriptions match its topic name. When a new subscription is established, the last retained message, if any, on each matching topic name must be sent to the subscriber.

C. Disadvantages

In the actual environment, there are always some accidents, such as hardware failure, signal interference, man-made and so on, leading to abnormal disconnection of the network, and then reconnected, that is, there are disconnected cases, which will bring about the following two problems.

Problem 1: Since the MQTT protocol states: If the server receives a retain message, it must discard any messages previously reserved for that topic; it will result in that if the publisher published more than one message during the disconnection of subscriber, the subscriber will only receive the latest message when reconnects to the MQTT server successfully, and the previous message will be missed.

978-1-5386-1803-5/18/$31.00 ©2018 IEEE

Problem 2: Since the MQTT protocol states that when a new subscription is established, it will be sent to this subscriber for each matching topic if there is a retained message on the server; it will result in that if subscribers reconnect frequently, and the publisher does not publish any new message, the subscriber will receive duplicate retain message every time it reconnects successfully. So, the subscriber needs to judge duplicate message and it may lead to subscriber exception

III. IMPROVEMENTS OF MQTT PROTOCOL

A. Improved Message Storage Model

The improved message storage model is shown in Table 2. Since the arrival of the messages is sequential, a chained storage structure is used, with each topic maintaining a list of retained messages.

TABLE II. IMPROVED MESSAGE STORAGE MODEL

Message Index (msg_index)	Message Content (msg_content)	Counts of Next Message to be Published (next_pub_count)	Cursor of Next Message (next_msg_cursor)

This model put forward two important variables: msg_index and next_pub_count. The msg_index is a unique identifier for marking a message and is used for locating the retained message list. When the server receives the message published by the publisher, the server generates a unique msg_index for the message. The next_pub_count is used to accurately delete the message that is no longer needed by any subscriber, and so that it can avoid the continually expand number of retained messages. When the next_pub_count reduce to 0, it indicates that the node has no effect and the node can be deleted directly.

B. Improved MQTT Control Packet

In order for the server to know which retain message has been received by the subscriber, the msg_index of the latest retain message received by the subscriber needs to be added to the control message. Therefore, the third bit of the first byte, which is used for reserved bit before, is used as the message filter identifier. The modified subscribe fixed header is shown in Table 3.

TABLE III. MODIFIED SUBSCRIBE PACKET FIXED HEADER

Bit	7	6	5	4	3	2	1	0
Byte 1	MQTT Control Packet Type				Message Filter Flag	Reserved		
	1	0	0	0	X	0	1	0
Byte 2...	Remaining Length							

When the message filter flag is 1, it indicates the payload contains message index part, and the modified SUBSCRIBE packet payload is shown in Table 4. When the message filter flag is 0, the payload is not modified. The improved payload adds a two-byte message index at the beginning. The message index is used to store the latest retain message index received by the subscriber, so that when the server receives the modified packet, it can find the retain message that the subscriber received last time according to the index, and then it can only send messages subscriber needed. Since the message storage mechanism designed in this paper only saves the messages that are not received during subscriber disconnection, in practice, it is almost impossible to reach 65,536 messages stored on the server under the same topic. Therefore, the design message index range is from 0 To 65535.

TABLE IV. MODIFIED SUBSCRIBE PACKET PAYLOAD

Description	7	6	5	4	3	2	1	0
Byte 1	Message Index (the range is from 0 to 65535)							
Byte 2								
Byte 3	Length MSB							
Byte 4	Length LSB							
Byte 5...N	Topic Filter							
Byte N+1	Reserved						QoS	
	0	0	0	0	0	0	X	X

TABLE V. MODIFIED PUBLISH PACKET VARIABLE HEADER

Description	7	6	5	4	3	2	1	0
Byte 1	Length MSB							
Byte 2	Length LSB							
Byte 3...N	Topic Name							
Byte N+1	1	Reserved						
Byte N+2	Message Index MSB							
Byte N+3	Message Index LSB							
Byte N+4	Packet Identifier MSB							
Byte N+5	Packet Identifier LSB							

In order to notify subscribers of the msg_index, it modifies the PUBLISH packet variable header sent by the server to the subscriber, the message index part is added after the topic name, and it is shown in Table 5. The message index part occupies at least one byte. The most significant bit in the byte is the message index flag, which is used to identify whether there is a message index or not. The lower seven bits complement 0 as the reserved bit. When the server sends a message to subscribers, if the message is a retain message, the message index flag must be set to 1, and the next two bytes should be added to store the corresponding message index. If not a retain message, the message index flag must be 0 and no other bytes needed. When subscribers receive the PUBLISH packet, they need to check the message index flag. If the flag is 1, subscriber should save the message index and related topic.

If the flag is 0, it means the message is not a retain message and no index need to be saved. When a subscriber subscribes to a topic, and if there is a message index saved under the topic, the message index must be filled into modified SUBSCRIBE packet payload, and the message filter flag of the fixed header must be set to 1. If there is no message index saved, the SUBSCRIBE packet is the same as the packet modified before and server only need to send the latest message to the subscriber.

IV. IMPROVED RETAIN MESSAGE HANDLING MECHANISM

A. Server Processing Mechanism

When server receives a PUBLISH packet, it needs to generate a message index for the retain message, and also needs to fill the message index into the PUBLISH packet variable header when sending the packet to subscribers. The detailed process is shown in Fig. 1.

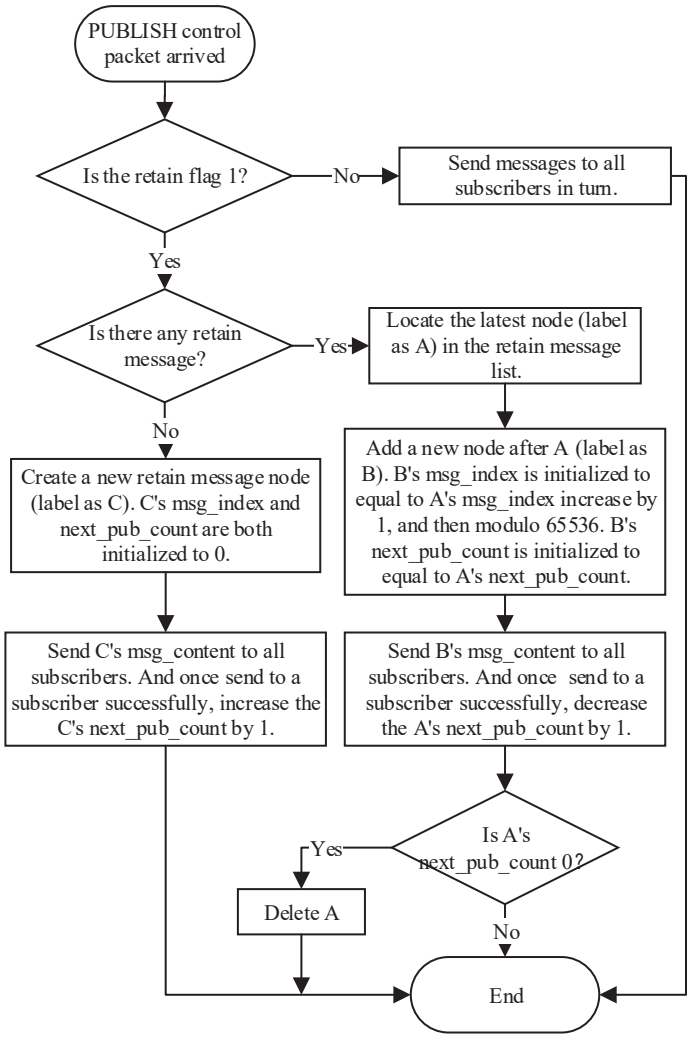

Fig. 1.Flow Chat of Handling Modified PUBLISH Packet in Server

When the server receives a SUBSCRIBE packet, if the message filter flag is 0, it indicates that the subscriber has never received the retain message before and it can only send the latest retain message. If the flag is 1, it indicates that the subscriber has received retain messages previously, and the server needs to send retain messages after the message whose message index is the same as msg_index parsed in the packet. The detailed process is shown in Fig. 2.

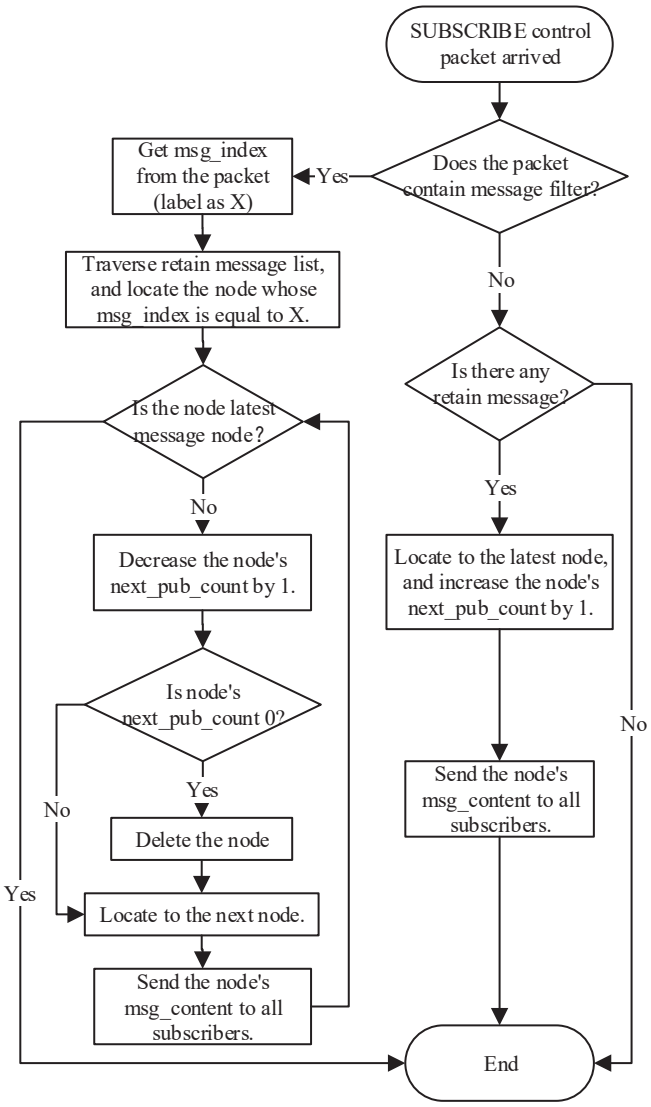

Fig. 2.Flow Chat of Handling Modified SUBSCRIBE Packet in Server

B. Client Processing Mechanism

When a subscriber receives a message, if the message contains a message index, the subject name and the corresponding message index are saved.

When subscribers subscribe to the topic, first determine whether there is a message index corresponding to the topic saved in the local, and if yes, send the message according to the modified subscribe message format; if not, send the message in the original message format.

The publisher's publishing mechanism does not need to be modified, because the message index is generated on the server side.

V. EXPERIMENT RESULT

A. System Environment

Server and client system environment is shown in Table 6.

TABLE VI. SYSTEM ENVIRONMENT

	Server	Client
CPU	Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz	Intel(R) Core(TM) i5-6300HQ CPU @ 2.30GHz
Memory	2GB	2GB
Hard Disk	33GB	14GB
Operating System	CentOS Linux release 7.3.1611 (Core)	Ubuntu 16.04 LTS

B. Experiment Results and Analysis

Corresponding to the previous two questions, designed two sets of tests: First, the publisher sends messages continuously in five minutes, and respectively, when the publisher publishes with different frequencies, and subscribers disconnect 10s, 5s, 1s, measure the message lost count. The experiment results are shown in Table 7. Second, the publisher does not publish messages and respectively measure duplicate message count under different reconnection times. The experiment results are shown in Table 8.

From Table 7, we can know that for the old retain message storage mechanism, the more frequently messages are sent and the longer the disconnection interval, the more messages are lost due to the disconnection. When the frequency of the messages sent is low or the reconnection interval is short, the message will not be lost. For the new retain message storage mechanism, because the server saves all retain messages, even if subscribers are disconnected, after the connection, the server will send all lost messages to subscribers, so the message loss rate is very low. The new retain message storage mechanism relative to the old may cost more memory, this is because under the new mechanism, the server will save all retain messages missed by subscribers and the old mechanism will save only one latest retain message.

From Table 8, we can know that, the old mechanism lead to 100% receive duplicate message, while the new mechanism solves the problem. This is because server knows what the subscriber needs according to the SUBSCRIBE packet under new mechanism. On this occasion, because no more retained message need to save, the memory usage is more stable.

TABLE VII. MESSAGE LOSS EXPERIMENT RESULT

Message sending Counts in One Minute	Total Number of Messages	Reconnection Interval (s)	Before Modification			After Modification		
			Not Received Number	Not Received Rate (%)	Memory Peak Usage in Server (%)	Not Received Number	Not Received Rate (%)	Memory Peak Usage in Server (%)
120	600	10	71	11.83	3.32	0	0.00	3.34
		5	32	5.33	3.32	0	0.00	3.33
		1	5	3.33	3.30	0	0.00	3.31
60	300	10	25	8.33	3.30	0	0.00	3.31
		5	12	4.00	3.29	0	0.00	3.29
		1	1	0.33	3.29	0	0.00	3.30
30	150	10	9	6.00	3.29	0	0.00	3.31
		5	5	3.33	3.28	3	2.00	3.30
		1	0	0.00	3.27	0	0.00	3.28
10	50	10	0	0.00	3.27	0	0.00	3.29
		5	0	0.00	3.26	0	0.00	3.27
		1	0	0.00	3.27	0	0.00	3.27
1	5	10	0	0.00	3.26	0	0.00	3.26
		5	0	0.00	3.26	0	0.00	3.27
		1	0	0.00	3.23	0	0.00	3.25

TABLE VIII. MESSAGE REPEATED RECEPTION EXPERIMENT RESULT

Subscriber Disconnection Times	Before Modification			After Modification		
	Repeat Number	Repeat Rate (%)	Memory Peak Usage in Server (%)	Repeat Number	Repeat Rate (%)	Memory Peak Usage in Server (%)
10	10	100	3.25	0	0	3.25
5	5	100	3.25	0	0	3.26
3	3	100	3.24	0	0	3.25
1	1	100	3.25	0	0	3.24

VI. CONCLUSIONS

This paper analyzes the disadvantages of MQTT retain message mechanism, and makes some corresponding improvements. The improved retain message storage mechanism can effectively avoid the message loss and repeated receiving problems. The experiment shows that the improvement effect is obvious.

Since the improved mechanism in this paper may store more retained messages, it will take up more memory. In practice, it should be based on the project requirements to determine whether to adopt this improved mechanism.

REFERENCES

[1] Banks A, Gupta R. MQTT Version 3.1.1 Plus Errata 01-OASIS Standard Incorporating Approved Errata 01[EB/OL]. (2015). http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[2] Liu F, Chen P, Jia J Y. Application of WebSocket and MQTT in Web Real-Time Communication System[J]. Computer Systems & Applications, 2016.

[3] Jin-Gang Y U, Geng Y F, Yang H B, et al. Design and Implementation of Message Engine Server Based on MQTT Protocol[J]. Journal of Chinese Computer Systems, 2016.

[4] Singh M, Rajan M A, Shivraj V L, et al. Secure MQTT for Internet of Things (IoT)[C]// Fifth International Conference on Communication Systems and Network Technologies. IEEE, 2015:746-751.

[5] Happ D, Karowski N, Menzel T, et al. Meeting IoT platform requirements with open pub/sub solutions[J]. Annales des Telecommunications, 2017, 72(1-2):41-52.

[6] Liu J X, Xiong X D, Jian-Dan F U. Push-Mode Services Invocation Based on Publish/Subscribe[J]. Computer Systems & Applications, 2012.

[7] Lee S, Kim H, Hong D, et al. Correlation analysis of MQTT loss and delay according to QoS level[C]// International Conference on Information NETWORKING. IEEE, 2013:714-717.

[8] Tantitharanukul N, Osathanunkul K, Hantrakul K, et al. MQTT-Topics Management System for sharing of Open Data[C]// International Conference on Digital Arts, Media and Technology. IEEE, 2017:62-65.

[9] ZENG Ang, LI Ning, YAN Jun. Research and improvement of Mosquitto file transfer. Computer Engineering and Applications, 2017, 53(4):123-127.