UDC 004.021:004.75

**QUALITY EVALUATION OF INFORMATION TRANSFER IN A DISPATCHING SYSTEM BASED ON MQTT ARCHITECTURE**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| ***V.F. Alekseev****Associate Professor, Department of Information Computer Systems Design, PhD of Technical sciences, Associate Professor**alexvikt.minsk@gmail.com*  | ***D.V. Likhachevsky****Dean of the Faculty of Computer Design of BSUIR, PhD of Technical Sciences, Associate Professor**likhachevskyd@bsuir.by* | ***G.A. Piskun****Associate Professor of the Department of Design of Information and Computer Systems of BSUIR, PhD of Technical Sciences, Associate Professor**piskunbsuir@gmail.com* |

***V.F. Alekseev***

*Graduated from the Minsk Radio Engineering Institute. The area of scientific interests is related to the development of methods and algorithms for constructing information and computer systems, the organization of educational and research processes at a technical university.*

***D.V. Likhachevsky***

*Graduated from the Belarusian State University of Informatics and Radioelectronics. The area of scientific interests is related to the study of problems of radio frequency identification of objects, the organization of educational and research processes at a technical university.*

***G.A. Piskun***

*Graduated from the Belarusian State University of Informatics and Radioelectronics. The area of scientific interests is related to the development of methods and algorithms for constructing information and computer systems, the organization of educational and research processes at a technical university.*

**Abstract.** The analysis of methods and algorithms for assessing the quality of information transfer, which ensure the quality of uninterrupted data transfer in a dispatch system built on the basis of the MQTT architecture, is carried out.

It is shown that tests can be developed to evaluate the operability of a dispatch system based on the MQTT architecture under various conditions of communication quality. The effectiveness of information transfer algorithms in a dispatch system based on the MQTT architecture with different levels of quality of service has been evaluated.

**Keywords**: MQTT architectures, information transfer quality assessment, scheduling, embedded systems.

**Introduction.** Generally, the process of transmitting information is carried out via the Internet. To connect to the server, various methods of transmitting information are used. An important criterion in this is the choice of network data transfer protocol. One of which is the MQTT protocol, which is predominantly used in embedded systems. The advantages of this protocol are that the quality and speed of the connection is not particularly important.

One of the main tasks performed by telemechanics and dispatching devices is uninterrupted monitoring of the condition of an object according to various indicators. Therefore, the creation and implementation of new dispatching and telemechanics systems is an urgent scientific task. The use of advances in modern systems engineering allows for more efficient and safe process control, as well as achieving the necessary control adaptability in the face of changes in the indicators for which tracking must be carried out, as well as changes in the tracking object.

Reliable operation of such systems is possible if reliable and timely information is available about the state of devices at the site, telemechanics and communications.

In this situation, issues related to the introduction of devices and new methods of obtaining and processing information become of great importance.

Assessing the quality of information transmission is one of the most promising and relevant areas of research for modern telemechanics and dispatching systems. This is due to the fact that when transferring data, high demands have been placed on the quality of uninterrupted data transfer (banking, medicine, agriculture, manufacturing, etc.).

A number of specialists [1-3] have analyzed methods and algorithms for the quality of information transmission in a dispatch and telemechanics system based on the MQTT architecture.

One of the disadvantages of similar systems is the poor quality of information transmission, as well as the loss of necessary data during information transmission over the network.

**Analysis of the principles of operation of the MQTT protocol.** *MQTT* (*MQ Telemetry Transport* is a messaging protocol that provides resource-constrained network clients with an easy way to distribute telemetry information. The protocol, which uses a publish/subscribe communication pattern, is used for communication between devices and plays an important role in the Internet of Things.

Additionally, on top of the *TCP layer* sits the standard *TLS security layer* (*Transport* *Layer* *Security*), formerly known as *SSL* (*Secure* *Sockets* *layer*). Port 8883 ensures communication security; if the broker's address works with this port, then traffic is transmitted with encryption.

This is a protocol designed specifically for *IoT*. An example of an *IoT* ecosystem using the MQTT protocol is presented in Figure 1.



*Figure 1.* Example of an *IoT* ecosystem using the *MQTT protocol*

Open and simple, it is designed to exchange information between different devices and modules. Simplifies the connection of communication channels quickly, efficiently and in a timely manner. Responsible for connection security, data transfer speed and practical functioning of systems and programs. Protects against all kinds of failures and malfunctions, performing its job efficiently.

resource - constrained IoT devices to send or publish information on a given topic to a server, which functions as an MQTT message broker. The broker then transmits the information to those clients who have previously subscribed to the client's topic. To a human, a topic looks like a hierarchical path to a file. Clients can subscribe to a specific level of a topic's hierarchy or use a wildcard to subscribe to multiple levels.

The MQTT protocol is a good choice for wireless networks that experience varying levels of latency due to occasional bandwidth limitations or unreliable connections. If the signing client's connection to the broker is lost, the broker buffers the messages and sends them to the subscriber when it reconnects. If the connection between the publishing client and the broker is disconnected without prior notice, the broker can close the connection and send subscribers a cached message with instructions from the publisher.

**Comparison of communication protocols in embedded systems. The** *MQTT* protocol is a simple messaging protocol that implements the publish/subscribe model *and is designed* to connect computerized devices connected to a local or global network with each other and various public or private web services.

The protocol was created to ensure openness, simplicity, minimal resource requirements, and ease of implementation.

In a network based on the MQTT protocol, there are 3 objects:

– publisher (*Publisher*) – *MQTT* client, which, when a certain event occurs, transmits information about it to the broker, publishing the corresponding topics;

– broker (*Broker*) – an MQTT server that receives information from publishers and transmits it to the corresponding subscribers; in complex systems, it can also perform various operations related to the analysis and processing of incoming data. Different brokers can communicate with each other if they subscribe to each other's messages;

– Subscriber *–* MQTT *client*, which, after subscribing to a broker, “listens” to it most of the time and is constantly ready to receive and process incoming messages on topics of interest from the broker.

*CoAP* protocol(*Constrained Application Protocol*) is a protocol developed by the Internet Engineering Task Force (*IETF*, *Internet Engineering Task* Force*)* and is described in *RFC* 7252. The protocol operates at the application layer, and is designed to transmit data over lines with limited bandwidth. *CoAP* was developed based on the *HTTP protocol*, is a binary version of it, but is not a blind compression of it. *CoAP* consists of a subset of *HTTP* functionality that has been newly designed to suit low power and low power consumption constrained embedded devices, such as indoor dust sensors. In addition, various mechanisms have been changed and some new features have been added to make the protocol suitable for the Internet of Things.

So, unlike the *HTTP protocol*, which is text-based and uses *TCP*, *CoAP*is a binary protocol that is transported over *UDP*, which reduces its overhead and increases flexibility in communication models. *CoAP* is organized into two layers: the transaction layer and the "*Request* / *Response*" layer.

**Testing of telemechanics and dispatching systems using various levels of quality of service.** In order to quantify the amount of data transferred when using the *MQTT protocol* with different *QoS parameters*, client-server transactions and the number of bytes transferred were analyzed. Table 1 contains information about the number of bytes and packets transferred per transaction. A transaction begins when the client sends data and ends when the server receives the data or, in some cases, when the client receives an acknowledgment.

Table 1. Number of bytes and packets transferred per transaction

|  |  |  |  |
| --- | --- | --- | --- |
| *QoS* | *MQTT* *QoS0* | *MQTT* *QoS1* | *MQTT* *QoS2* |
| Number of bytes | 75 | 135 | 255 |
| Number of packages | 1 | 2 | 4 |

The message is divided into two parts: useful information and service information. These parts affect the cost of channel resource and battery energy. To improve efficiency, a reduction in overhead information is required. Table 2 shows the ratio of service information to useful information as a percentage when transmitting one message.

Table 2. Ratio of useful information to service information

|  |  |  |  |
| --- | --- | --- | --- |
| *QoS* | *MQTT* *QoS0* | *MQTT* *QoS1* | *MQTT* *QoS2* |
| Helpful information, % | 16.8 | 16.5 | 16.5 |
| Service information, % | 83.2 | 85.5 | 85.5 |

In *MQTT* with *Qo S 0*, the service fields in the packet occupy a small volume, so a small amount of energy is spent during a communication session.

Figure 2 shows the results of a study of the delay in message transmission.

*Figure 2.* Delay in milliseconds for each message

Table 3 shows the percentage of lost messages. Data analysis shows that despite the fact that when using *QoS1* and *QoS2* the delay in sending messages significantly increases, the percentage of lost messages decreases. If you do not take this fact into account, you can suffer significant packet loss, which can be critical in some systems. In other systems, where each message is not so important, you can get a significant increase in the time for message transmission, which can be critical with a large number of small messages

Table 3. Percentage of lost messages

|  |  |  |  |
| --- | --- | --- | --- |
| Message size, *B yte* | Lost at *QoS0*, % | Lost at *QoS1*, % | Lost at *QoS2*, % |
| 150 | 3.00 | 1.00 | 0.00 |
| 200 | 5.00 | 1.30 | 0.04 |
| 250 | 6.00 | 1.50 | 0.05 |
| 300 | 6.70 | 1.80 | 0.05 |

**Principles for achieving high quality communications when using the *MQTT data transfer protocol*.** A set of principles can be proposed that help optimize the use of bandwidth and data:

1. Choosing the right *QoS*. One of the key features offered by *MQTT* is quality of service (*QoS*). *QoS0* messages are the simplest, also called fire and forget messages. These messages have no acknowledgment from the broker (but still have acknowledgment from the *TCP layer*) and therefore are not guaranteed to be delivered. *QoS1* messages are guaranteed to be delivered, although it is possible that they may be delivered multiple times. *QoS1* messages involve two layers of application-level communication. Messages of type *QoS2* are guaranteed to be delivered exactly once. *QoS2* messages have the highest overhead.

Given the fact that *QoS level* and overhead are inversely proportional, the strategy for choosing the *QoS level* for your messages is quite simple. All high-frequency data (usually real-time data) can be sent using *QoS0*, since losing a few data packets may not be critical. *QoS1* should be used for messages that require guaranteed delivery, mainly events and commands.

When developing any application, it is important to objectively understand the cost of *QoS choices*. As an example, the data consumed by a message was measured at different *QoS levels*. The " *HelloWorld "* message was posted to the " *test\_test* " thread with three different *QoS levels*, and the data exchanged was captured using *Wireshark* (Table 4).

Data consumption is reduced by approximately 50% when using *QoS1* compared to *QoS2*. Likewise, *QoS0* uses 40% less data than *QoS1*.

Therefore, it is correct to use *QoS0* for periodic information that needs to be sent to the server. You should not use *QoS1* for commands from the server because there is the possibility of redundant command delivery. Although *QoS2* seems like an obvious solution for commands from the server, it is not a cost-effective choice.

Table 4. Number of bytes consumed by a message at different *QoS levels*

|  |  |  |  |
| --- | --- | --- | --- |
| *QoS* | *MQTT* *QoS0* | *MQTT* *QoS1* | *MQTT* *QoS2* |
| Number of bytes | 87 | 126 | 241 |

2. Minimize *QoS2 messages*. *QoS2* message overhead is comparable to *HTTP*. They have 50% more overhead than *QoS1 messages*. *QoS1* messages can be used to replace *QoS2 messages* in most applications. The problem with *QoS1* messages is that they can be delivered multiple times. There are two ways to avoid this.

The first solution applies when you control the implementation of the *MQTT client library*. Each *MQTT publishing package* has a "Package ID" field. This field is typically increased for each new package published by the broker to the client. If you receive the same message (in case of *QoS0*), then the Packet ID field will remain the same. The client can maintain a list of "packet IDs" received in the last few messages (~10) and use it to determine whether a new message received is a duplicate.

If you don't have control over the implementation of the *MQTT client library*, you can create a Virtual Package Identifier (*VPI*) mechanism. The payload of each *MQTT message* can be configured to contain *a VPI*, this *VPI* can be incremented by the message sender whenever a new message is published. The client can maintain a list of the last few *VPIs received* and use it to filter out duplicate messages.

3. Careful choice of topic names. The title field contains the topic name in *UTF8 format*. This means that the long topic title makes up the majority of the message.

In the event that it is necessary to publish (*QoS0*) information of 20 bytes on a topic. Post message size will include post size, title size, subject size. Thus, the size of the topics take up a significant portion of the message. You should choose a naming strategy so that they contain only important information.

**Conclusion.**Data and bandwidth usage is a major constraint in the design of *IoT edge devices*. Unlike cellular data for general use, enterprise-grade cellular data with long uptime is typically expensive. Therefore, data consumption limits become more relevant if the peripheral device relies on cellular data. Poor application design can prevent you from taking advantage of the low data consumption that *MQTT* offers for *IoT systems*.

**Reference list**

[1] Roslyakov A.V., Vanyashin S.V., Grebeshkov A.Yu. Internet of Things: tutorial. Samara: PGUTI, 2015. – 200 p.

[2] Martz N, Warren D. Big data. Principles and practice of building scalable real-time data processing systems. M.: Williams, 2016. – 368 p.

[3] Taylor J, Rayden T. Gaining competitive advantage by automating hidden decisions. M.: Symbol-plus, 2009. – 448 p.

**Authors’ contribution**

**Viktor Alekseev** – led the research on evaluation the quality of information transmission in a dispatch system based on MQTT architecture.

**Dmitry Likhachesky** – statement of the research problem, description of the operating principle of the MQTT protocol, comparison of interaction protocols in embedded systems, analysis of the results obtained.

**Gennady Piskun** – testing the telemechanics and dispatching system using various levels of quality of service, describing the principles of achieving high quality communications when using the MQTT data transfer protocol, forming the structure of the article.