FEATURES OF APPLICATION OF DATA TRANSMISSION PROTOCOLS IN WIRELESS NETWORKS OF SENSORS

Olexander Belej
Department of Computer-Aided Design
Lviv Polytechnic National University
Lviv, Ukraine
Oleksandr.I.Belei@lpnu.ua

Natalia Nestor
Department of Computer-Aided Design
Lviv Polytechnic National University
Lviv, Ukraine
natalia.nestor@gmail.com

Orest Polotai
Department of Information Security Management
Lviv State University of Life Safety
Lviv, Ukraine
orest.polotaj@gmail.com

Jan Sadeckii
Institute of Computer Science
Opole University of Technology
Opole, Poland
j.sadecki@po.opole.pl

Abstract—This article discusses the vulnerabilities and complexity of designing secure IoT-solutions, and then presents proven approaches to protecting devices and gateways. Specifically, security mechanisms such as device authentication (including certificate-based authentication), device authentication, and application a verification of identification are described. The authors consider a protocol of message queue telemetry transport for speech and sensor networks on the Internet, its features, application variants, and characteristic procedures. The principle of "publisher-subscriber" is considered. An analysis of information elements and messages is carried out. The urgency of the theme is due to the rapid development of "publisher-subscriber" architecture, for which the protocol is most characteristic.

Keywords—protocol, messages, protecting devices, authentication, sensor networks, data

I. INTRODUCTION

The protocol of message queue telemetry transport (MQTT), for the time being, has gained popularity and has become a de facto standard in projects designed to create solutions for the Internet of Things (IoT). By Internet, things are understood as a set of various devices, sensors, actuators, connected to the network by means of any available communication channels, using different protocols of interaction with each other and a single protocol of access - IP to the global network Internet. In this case, the private tasks of Machine-to-Machine (M2M) machine interactions are already being effectively dealt with, including the level of connection of these systems to the Internet, both for the creation of industrial automation systems, for example, for the construction of "smart home" systems. It is obvious that critical systems require the presence of a local arbitrator or a broker and devices that allow you to work out a solution to the situation, regardless of the quality of the connection to the Internet, as well as in the event of a complete break of communication.

The interconnection of the hardware platform, messaging protocols and their software implementation was the starting point in the rapid progress of the development of modern automation systems. Next, let's look at hardware solutions and software components for the implementation of one of the most successful protocols for messaging in technical systems.

The Internet of Things (IoT) represents tremendous opportunities for organizations and consumers, especially in such areas as healthcare, warehousing, transportation, and logistics. With the widespread use of the Internet of things, developers face new challenges - they need to guarantee sufficient security for IoT applications since these applications manipulate a large amount of confidential data. IoT solutions have already identified many security breaches, so developers need to pay great attention to integrating protection into IoT applications in the design and implementation of such solutions.

Wireless Sensor Network (WSN) allows receiving data for stand-alone systems, thanks to its small size, simple layout and wireless transmission. In researching [1], key factors were distinguished in the design of autonomous systems based on WSN, the basic structure and scheme for data transmission, focused on the collection and transmission of data on the electric autonomous system, is proposed. In the study [2] proposed a protocol for data transmission between measuring sensors containing buffered data about data packets to be transmitted to other sensors. Such transfer of packets is carried out by means of an appropriate relay selection algorithm that reduces power loss.

When transmitting data between sensors in the WSN, the corresponding communication protocol is used through the network protocols TCP/IP and RFC. The functionality of such a system in [3] was tested in a network with four sensors that measure the temperature around the heat source. The NoPSM protocol can quickly and accurately design interconnections between interfaces, as well as the transmission of block data without touching the normal network [4]. This protocol allows you to decide on the parity of transmission with unlimited parameters that allow you to evaluate the quality of any active connections after initiating a new link and improving the bandwidth obtained from simultaneous transfers.

The study [5] examines the use of wireless sensor networks for the mechanical diagnosis of industrial monitoring systems. The proposed algorithm uses a data clustering protocol and each cluster's vertex is modeled by several layers of data. The proposed protocol in [6] is used to increase the density of the network based on the given threshold values. It also allows each node in the WSN to automatically identify the attributes of neighboring channels. In [6], it is achieved by switching channels to achieve maximum data transmission.

In [7], modification was proposed based on the stabilization of sensor nodes, taking into account the energy coefficient. The purpose of this modification is to calculate distance parameters when determining the threshold value that will be used to form new extended clusters [7]. The problem of dynamic optimization of the encryption and data transfer strategy is to minimize the deviation function in the source representation of the system in accordance with the limits of the stability of the data queue. It is formulated for time-varying channels and sources. In [8] is proposed to apply the Lyapunov method based on perturbation, which is close to optimal, since it has clear and controlled boundaries between the optimal rupture and the size of the queues.

WSN sensors are powered by batteries that are commonly used at the device level. [9] proposed an extended Green MAC protocol for the WSN server. The proposed protocol allows remote sensing of data about network events to be managed. Data transmission is carried out using the proper power selection algorithm. The simulation results presented compare the performance of an existing protocol with the Green MAC Energy Management Protocol developed.

Wireless Sensor Networks (WSNs) provide a new level for collecting system data about various features of the network with its small-sized features, simple layout and wireless transmission. The paper [10] considers key factors in designing WSN-based IoT devices, proposes a WSN-based structure and data transfer scheme that targets the transmission of electrical energy between devices and defines the development of appropriate modules.

Energy efficiency is a major requirement for the Internet of things, as it is expected that many sensors will be completely autonomous and able to work for years without replacing the battery. Data archiving is aimed at saving energy by reducing the amount of data transmitted over the network, but also affects the quality of the information received. The paper [11] formulates the optimization problem for developing a strategy for encoding and data transmission. Here, optimal offline policies are proposed for the distribution of energy parameters and transmission over a network with a TDMA-based dynamic access scheme.

The paper [12] proposes an energy-efficient E-MAC protocol, which is capable of generating any duty cycle based on its load on the motion. The results of this simulation confirm that E-MAC exceeds existing energy efficiency protocols in terms of power consumption, network bandwidth and network lifetime. The proposed energy saving protocol allows you to generate an arbitrary boot loop based on the traffic load of the sensor node, instead of switching to sensor sleep mode to reduce power consumption.

Today, the Internet of Things covers a huge range of industries, ranging from industry and ending with food. Data from IoT can be transmitted to the network of general communications and stored on a cloud server. The protocols used to transfer data to the IoT now number about twenty five. Among the existing Internet protocols, MQTT, CoAP and HTTP/2 protocols are the most commonly used to collect and transfer data between devices and the IoT server infrastructure. However, each protocol has its own peculiarities of functioning, which are manifested in the emergency modes of the sensor network. For this reason, the choice and use of any protocol has become a topical issue, as the load on such networks increases over time. In the proposed study, the features of the functioning of the MQTT, CoAP and HTTP/2 protocols based on WSN were considered and analyzed. Our research covers the use of special protocols for various types of sensory devices and networks.

II. Construction of a wireless touch-based network based on the MQTT-protocol

Any devices on the network may be exposed to unauthorized access and external influence. An aggregate of data from measuring devices may be of interest to hackers and thieves, so attacks on IoT computer networks can damage physical devices and services. The protection of software applications of IoT networks is important both for the reputation of the enterprise and for the well-being of the users themselves of the products and services.

Clients and owners of IoT solutions understand the need to protect such applications, so the development and implementation of security tools create new opportunities for implementing creative potential of developers. When developing the majority of network applications, IoT owners are constantly forced to seek a compromise between security and ease of use.

Very often, IoT devices do not have the necessary computing power and memory space to use sophisticated cryptographic algorithms that make their protection more secure than unauthorized access.

The IoT application consists of several basic levels: devices and gateways; network software applications. All components of IoT application at each level should have adequate security measures to protect against various vulnerabilities. The level of IoT software applications is most open to cyber-attacks. This level consists of applications that consist of web applications, cloud services, mobile applications. These programs can be installed on or working with IoT devices.

The security of the IoT application program is an important and integral part of all stages of their life cycle. At the stage of designing and developing the program part, it is necessary to really evaluate the capabilities of the proposed security and the requirements developed, which will ensure the limited use and confidentiality of the data.

On Fig. 1 shows three levels of software application for IoT based on the principles of IBM Watson IoT Platform. At the network level and database levels, data transfer and storage is realized using IBM Bluemix cloud technology.

The protocol of message queuing telemetry transport (MQTT) itself specifies only a small number of protection mechanisms, but all its common implementations support modern security standards. The MQTT protocol does not require a special approach to protect applications based on it, leaving this task to the discretion of the software application developer. This approach allows us to build a security system and build requirements for it within the framework of the program implementation of the IoT.

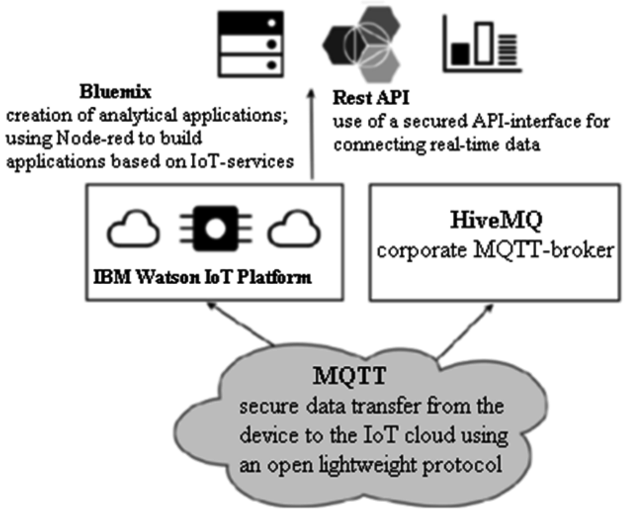

Bluemix
creation of analytical applications; using Node-red to build applications based on IoT-services
Rest API
use of a secured API-interface for connecting real-time data
IBM Watson IoT Platform
HiveMQ
corporate MQTT-broker
MQTT
secure data transfer from the device to the IoT cloud using an open lightweight protocol

Fig. 1. The structure of the wireless sensor network as an IoT-application based on the protocol of Message Queuing Telemetry Transport.

For implement of security MQTT is used the transport layer security (TLS). This allows you to encrypt the data transmitted, while maintaining their integrity. In most cases, MQTT protocols require authorization on the server to control access to databases and data systems.

In addition to the computer IoT model of the device, we offer a client application that reflects the spam messages in the wireless sensor network, which are transmitted between devices.

In MQTT authentication is part of the protection level of the transport level and the level of applications. At the transport level, TLS guarantees client authentication on the server by using client certificates and authenticating the server to the client by validating the server certificate. At the application level, the MQTT protocol provides authentication by user name and password.

Developers can use multiple approaches to ensure proper registration and identification of the device. The choice of approach depends on the requirements to protect the solution and the capabilities of the device to implement the appropriate approach.

III. THE FORMATION OF A PROTECTION MECHANISM MQTT-PROTOCOL FOR DATA TRANSMISSION BETWEEN SENSORS AND CLOUD SERVICES

To implement the security of the network and application levels, we authenticate the users in the MQTT protocol. At the network level, TLS guarantees user authentication. The use of user certificates and authentication on the server is implemented through the verification of server certificates. At the MQTT protocol level, we provide authentication by entering a username and password.

To ensure IoT device registration in a software broker, developers use several approaches that depend on the security and features of the device.

The IoT device authenticate through the MQTT protocol, enter the user names and password into the corresponding fields of the CONNECT message. When connecting to the MQTT software broker, the client must send a username and password. In this case, the user name is encoded in the UTF-8 format, and the password is a set of characters in binary format. If network encryption is not used, then the MQTT protocol does not encrypt the username or password and the message is sent as plain text.

In case of successful access permission, the user can be authenticated in the software broker using the password field in the CONNECT message. After that, the user name becomes an information line to determine the truth. In this case, the maximum password size in the MQTT protocol is 65535 bytes and the length of the identifier cannot exceed these limits.

The programing broker can use this token to perform various checks, including the following: checking the token signature; checking the validity of this token for its expiration; check the authorization server for cancellation of this token.

When an IoT device is connected to the MQTT protocol software broker, checks may be used to identify and use software applications. When identifying and using a software broker, you also need to authorize the application. The user's authorization can be made in several ways: the identifier includes authorization of the user within the query to the database; there may be an access intermediary to the databases in which the user's identity is implemented.

IBM Watson IoT Platform software applications may have user ID, access key, and application software authenticator. The IoT application key identifier is generated when registering an IoT device. They can be used when connecting to the IBM Watson IoT Platform software broker.

The protection of the IoT device is implemented through its validation in the register of trusted devices. This allows the broker to "trust" the software application of the IoT device, which sends management commands. Consider below the various types of data protection to form a security system that you can use to establish such "trust".

In addition to authentication mechanisms that are provided in the MQTT protocol, additional protection and device identification tools can be introduced for software applications of the IoT device. We consider the approach to implementing a one-time password authentication (OTPA) user of the IoT device. OTPA can be a reliable means of protecting your device from inappropriate use, eliminating the risk of unauthorized access to it.

When OTPA starts exchanging data with the IoT application, only users who have authenticated can start the device after the device is started. Not all IoT devices may have keyboard input, we can implement a simple property switch to activate this protection scheme, depending on the IoT device being used. If OTPA is enabled, the device sends an OTPA request to the software broker of the IoT device. This is done by the usual MQTT protocol messaging. The corresponding flow is shown in Fig. 2.

The MQTT protocol software brokers used to realize the capabilities of IoT devices in enterprise systems support the certification of IoT devices that the broker can use as part of the identification process. Such identification is taken into account in software applications where security requirements are very stringent and devices can initiate certificates.

We will use HiveMQ to demonstrate the two-way authentication of the Secure Sockets Layer (SSL).

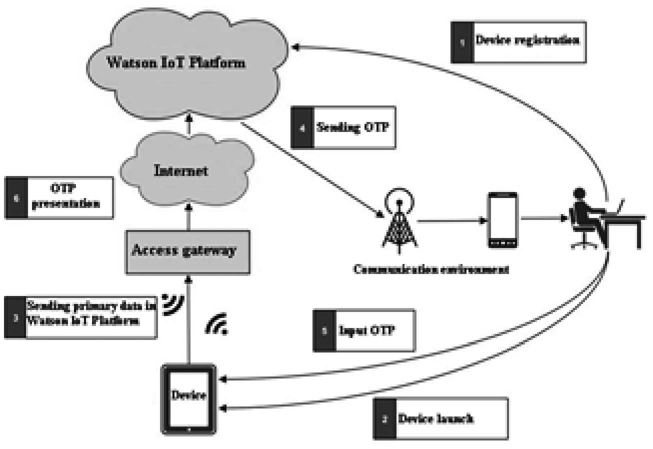


Fig. 2. Transferring data between cloud storage devices using the one-time password authentication.

Only the standard MQTT protocol tools are used to interface the device. Certificate identification provides a high level of security for IoT software applications. Life-cycle management of such certificates for several IoT devices is quite expensive.

The IBM Watson IoT Platform service we reviewed does not support the identification of user certificates. To use TLS, the server must have an open and private key at a time. When identifying users in TLS, the X509 certificate is checked before establishing a secure connection.

Users can also have a unique pair of public and private keys to implement identification in the TLS protocol. After verifying the server certificate, the user sends his certificate as part of the TLS identification procedure. If the user certificate cannot be verified, then the server may cancel the network data transfer procedure. This approach allows you to identify the user before setting up a secure connection.

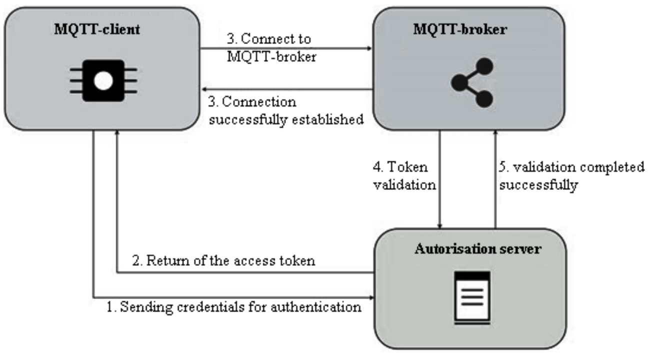
The implementation of custom certificates allows you to verify the identity of the MQTT users of the protocol, to authenticate them at the network level and to block invalid users before sending the CONNECT message to the MQTT protocol.

When using custom certificates, you can only install a secure connection for trusted users. In this case, resources are stored in a software broker, which is convenient when using expensive identification tools in the MQTT protocol. User identification is implemented through the TLS network protocol to establish a network connection.

All messages in the MQTT protocol are published with an identified theme and have an appropriate filter to include wildcard characters. Most MQTT-based servers have permission to publish and subscribe to a topic.

All IoT devices operate on the same network with a definite set of devices. Custom data after identification allows you to determine which device from the IBM Watson IoT Platform service will be linked to topics with the built-in network. This configuration prevents IoT devices from functioning as other devices. The only way to operate as another device is to get the account of the corresponding IoT device.

To use our authoring tool on the MQTT server, the protocol for the devices can use the OAuth protocol environment. The OAuth 2.0 protocol allows you to separate the authorization server from the resource server. When using OAuth 2.0, the user passes its credentials to the authentication server, which verifies the authenticity and returns access rights with permission to access the server. After verifying the access rights, it is used to connect to the server with the MQTT protocol. The MQTT server checks access rights and gives access to the server.



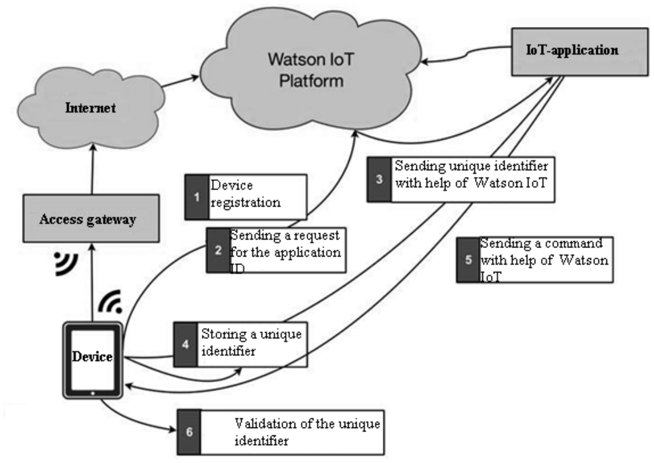
MQTT-client
MQTT-broker
3. Connect to MQTT-broker
3. Connection successfully established
4. Token validation
5. validation completed successfully
Autorisation server
2. Return of the access token
1. Sending credentials for authentication

Fig. 3. The server of MQTT protocol checks access token.

An additional level of protection between software applications of IoT devices is the verification of the program ID. It is designed to prevent any fraudulent program from sending commands to the device. This tool is used both for protection on startup and for security of communication on the network. In this case, the IoT stores a unique application identifier and checks it when executing commands from the IoT device application.

The IoT software application sends an invalid unique identifier to the command and the device ignores this command. When saving information in a software broker unique identifier of the program IoT can be stored in it in encrypted form. In this case, after each reboot, we do not have to constantly ask for a unique ID.

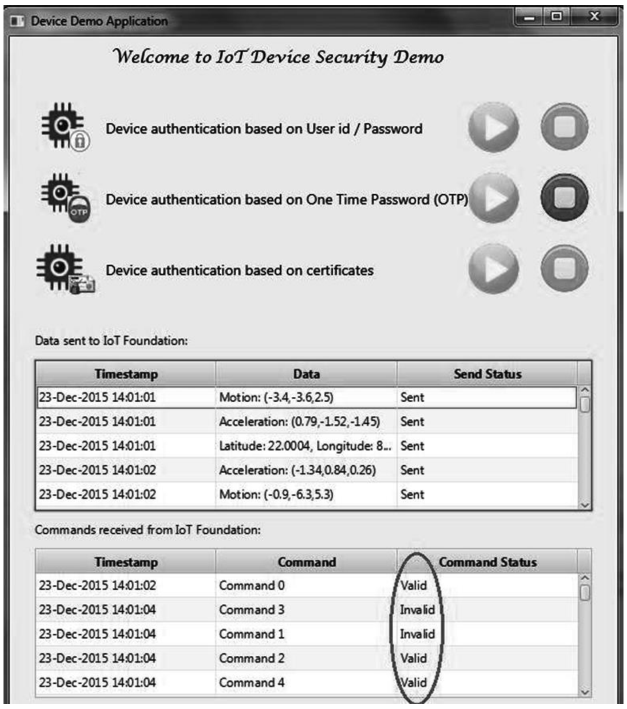
If the program ID verification is activated and the unique identifier store is activated, then the software application of the IoT device tries to recover a unique identifier from the encrypted file. If a device cannot load a unique application ID, then it creates a request for a unique application ID.



Watson IoT Platform
IoT-application
Internet
1 Device registration
3 Sending unique identifier with help of Watson IoT
Access gateway
2 Sending a request for the application ID
5 Sending a command with help of Watson IoT
Device
4 Storing a unique identifier
6 Validation of the unique identifier

Fig. 4. Checking the application of identification between the IoT application and device.

The following figure shows the screen of a computer model of the IoT device using a software application identifier. The unique ID of the IoT device coming with the command is checked for coincidence with the stored application ID, and the corresponding command is executed or ignored.


Device Demo Application

Welcome to IoT Device Security Demo

Device authentication based on User id / Password

Device authentication based on One Time Password (OTP)

Device authentication based on certificates

Data sent to IoT Foundation:

Timestamp	Data	Send Status
23-Dec-2015 14:01:01	Motion: (-3.4,-3.6,2.5)	Sent
23-Dec-2015 14:01:01	Acceleration: (0.79,-1.52,-1.45)	Sent
23-Dec-2015 14:01:01	Latitude: 22.0004, Longitude: 8...	Sent
23-Dec-2015 14:01:02	Acceleration: (-1.34,0.84,0.26)	Sent
23-Dec-2015 14:01:02	Motion: (-0.9,-6.3,5.3)	Sent

Commands received from IoT Foundation:

Timestamp	Command	Command Status
23-Dec-2015 14:01:02	Command 0	Valid
23-Dec-2015 14:01:04	Command 3	Invalid
23-Dec-2015 14:01:04	Command 1	Invalid
23-Dec-2015 14:01:04	Command 2	Valid
23-Dec-2015 14:01:04	Command 4	Valid

Fig. 5. The screen with the computer model of IoT device.

The software application of the IoT devices sends a unique identifier to the device after receiving the authorization request. After authorization, the IoT stores the program ID in the memory and checks it in each command from the IoT devices to determine the match.

Flexible multi-level protection and encryption options allow users to secure data transfer and protect their confidentiality from unauthorized access. In other variants, users prefer the protection based on special encryption through SSL in the absence of secure data sharing.

Software applications of the IoT devices can use a special encryption to send messages over the network. The IBM Watson IoT software broker considered by us does not support special encryption, but this encryption is supported by the software broker HiveMQ.

IV. Discussion

In order to quantify the amount of data transmitted using protocols, client-server transactions and the number of bytes transmitted were analyzed. Table 1 shows the number of bytes and packets transmitted per transaction for MQTT, CoAP, and HTTP/2. A transaction begins when the customer sends the data and ends when the server receives data or, in some cases, when the customer receives the confirmation.

TABLE I. Bytes sent in a single client-server transaction

Protocol	MQTT – QoS0	MQTT – QoS1	MQTT – QoS2	CoAP	HTTP/2
Bytes per transaction	75	135	255	162	1149
Single Transaction Packages	1	2	4	2	10

As shown in Table 1, the HTTP/2 transactions, although using the HPACK header compression technology, still includes significantly more bytes and packets. The MQTT and CoAP protocols have a short header length. But CoAP at the transport layer uses UDP, so it has a smaller packet size, unlike MQTT. After encapsulation in TCP and UDP layer headers, the MAC packet of these protocols can be transmitted in one MAC frame, which is 80 bytes in size.

TABLE II. Attitude of useful information to service information in one message

Protocol	MQTT – QoS0	MQTT – QoS1	MQTT – QoS2	CoAP	HTTP/2
Useful information, %	16,8	16,5	16,5	15,3	10,9
Service information, %	83,2	85,5	85,5	84,7	89,1

The message is divided into two parts: useful information and service. These parts affect the cost of channel life and battery power. To improve efficiency, a reduction in service information is required. Table 2 shows the ratio of service information to users as a percentage when sending a single message.

The basic level of trust "delivery maximum" (QoS0) requires a separate channel with guaranteed delivery of data with preservation order. The client or server executes the command Pub (package) and forgets about it, no additional checks, nothing at all. The level of trust at least "one time delivery" (QoS1) requires confirmation of receiving a special package by transferring each received package. The highest level of trust is "just after delivery" (QoS2) with double confirmation on both sides of the symmetrical Publish packages.

The CoAP and MQTT protocols with Qos0 service fields in the packet occupy a small amount, so a small amount of energy is spent during a communication session. Internet Things mainly transmit data over the air, in our case via WiFi, so the problem of power supply will be important to increase the life cycle of the device. For CoAP, you can set the server to work in such a way that a request is created when the data is updated, and the sensor sends new values. Thus, it is possible to avoid constant data transfer and inefficient energy costs. This option is suitable for devices with limited resources.

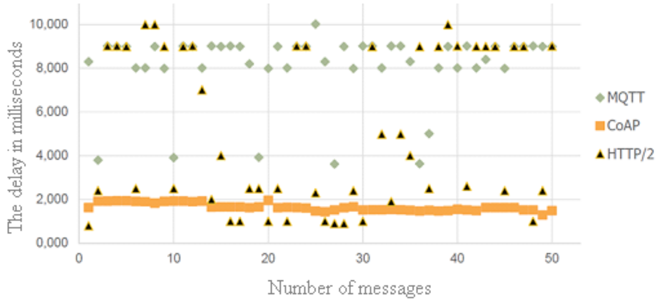


Fig. 6. Delay in sending data from client to server.

In Fig. 5 shows the results of an experimental study of the magnitude of the delay in the transmission of messages.

It can be seen that the CoAP protocol has a stable small delay. CoAP uses the UDP transport layer protocol, which allows you to quickly process data and transmits packets of small length, which reduces redundancy in the transmission channel. However, there are situations when the client-server connection is not stable. Usually, this situation occurs when, over a period of time - a few hours, the client sends data from messages of different lengths. In Fig. 6 shows the dependence of packet loss depending on their size.

The graph shows that with increasing message size, the percentage of packet loss increases. It is worth noting that the CoAP protocol is more likely to be lost than the MQTT and HTTP/2 protocols, which is due to the use of the UDP transport protocol, which does not guarantee the delivery of messages. It is also worth considering that we consider the MQTT protocol for 3 different types of quality (QoS0, QoS1, QoS2). According to the quality types, we see that the greater the degree of reliability we assign to a message (QoS2), the lower the probability of loss.

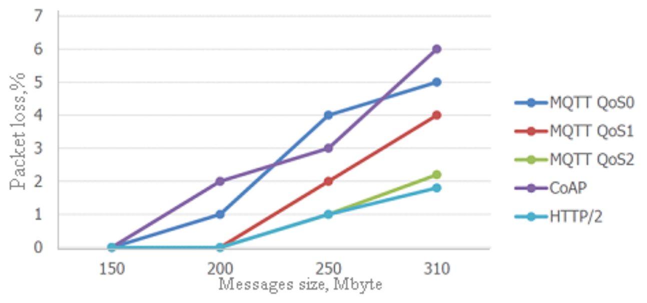


Fig. 7. Graph the effect of message size on packet loss.

The CoAP and MQTT protocols allow us to connect the gateway to the server. Currently, a large number of protocols are used for these purposes, but these protocols are most commonly used in the implementation of the IoT. Using CoAP and MQTT protocols can be effective for sending short messages. The HTTP/2 protocol is better to use for WEB of Things (WoT).

V. Conclusion

To choose the approach to authenticating an IoT device, you need to know clearly the solution and the array of data with which this decision will be made. Therefore, we need to use secure networks and gateways for the transmission of confidential data. At the same time, authentication, identification and secure communication are very important. You can use one-time passwords to implement additional protection and improve the physical security of your device. Based on the software application's capabilities, you can use the user authentication of the network to support external vendors of authorization. Authorization for network access must be strictly controlled to block unauthorized access to subscriptions and databases. Therefore, in our opinion, you should enter a limit on the size of network messages so that the user could not block the program application itself IoT.

The article describes the most popular protocols of the Internet of Things CoAP, MQTT, HTTP/2, which are used to send information from the sensor to the cloud server. The study revealed that the MQTT and CoAP protocols are characterized by lower data transfer overhead (due to a small amount of service traffic) and lower bandwidth than the HTTP / 2 protocols. These protocols are well adapted for low-power devices of the IoT based on microcontrollers. For its work, the MQTT protocol does not require a permanent connection between the client and the server, as well as the CoAP protocol, which is not to say about HTTP/2. Experimental results have shown that the effectiveness of the considered protocols depends on various conditions of the communication network. The most optimal is the MQTT protocol, in which it is possible to set parameters responsible for the reliability of message delivery. The HTTP/2 protocol provided that the communication network works stably will be more suitable for Web Things - information is delivered quickly and can be visualized in mobile applications or on personal computers. In this regard, at the present time, the correct choice of protocols for various IoT helps to solve the problem of saving resources, both energy consumption and guaranteed delivery. This is especially true in connection with the increase in the number of IoT devices.

References

[1] V. Ababii, V. Sudacevschi, M. Podubnii, and I. Cojuhari, "Sensors network based on mobile robots," 2014 International Conference on Development and Application Systems (DAS), Suceava, 2014, pp. 70-72.

[2] L. M. Varalakshmi and M. Preethi, "Performance enhancement of green MAC power saving protocol for corona-based wireless sensor networks," 2016 10th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, 2016, pp. 1-6.

[3] V. Ababii, V. Sudacevschi, M. Podubnii, and I. Cojuhari, "Sensors network based on mobile robots," 2014 International Conference on Development and Application Systems (DAS), Suceava, 2014, pp. 70-72.

[4] H. Chen, Z. Zhang, L. Cui and C. Huang, "NoPSM: A Concurrent MAC Protocol over Low-Data-Rate Low-Power Wireless Channel without PRR-SINR Model," in IEEE Transactions on Mobile Computing, vol. 16, no. 2, pp. 435-452, 1 Feb. 2017.

[5] Q. Chen, Y. Hu, J. Xia, Z. Chen and H. Tseng, "Data Fusion of Wireless Sensor Network for Prognosis and Diagnosis of Mechanical Systems," 2017 International Conference on Information, Communication and Engineering (ICICE), Xiamen, 2017, pp. 331-334.

[6] T. Hassan, S. Aslam, and J. W. Jang, "Fully Automated Multi-Resolution Channels and Multithreaded Spectrum Allocation Protocol for IoT Based Sensor Nets," in IEEE Access, vol. 6, pp. 22545-22556, 2018.

[7] G. Kaur, R. Bhatti and P. Kaur, "E-CHATSEP: Enhanced CHATSEP for clustered heterogeneous wireless sensor networks," International Conference on Computing, Communication & Automation, Noida, 2015, pp. 403-407.

[8] C. Tapparello, O. Simeone and M. Rossi, "Dynamic Compression-Transmission for Energy-Harvesting Multihop Networks With Correlated Sources," in IEEE/ACM Transactions on Networking, vol. 22, no. 6, pp. 1729-1741, 2014.

[9] L. M. Varalakshmi, M. Preethi. "Performance enhancement of green MAC power saving protocol for corona-based wireless sensor networks", 2016 10th International Conference on Intelligent Systems and Control (ISCO), pp. 129-134, 2016.

[10] M. Qiu, P. Jiang, Q. Chen, Y. Jin. "Application study on prognostics and health management of armored equipment based on wireless sensor networks", 2014 Prognostics and System Health Management Conference (PHM-2014 Hunan), pp. 216-121, 2014.

[11] C. Pielli, A. Biason, A. Zanella, M. Zorzi. "Joint Optimization of Energy Efficiency and Data Compression in TDMA-Based Medium Access Control for the IoT", 2016 IEEE Globecom Workshops (GC Wkshps), pp. 89-96, 2016.

[12] C.-M. Chao, C.-H. Jiang. "Energy Efficient Protocol for Corona-Based Wireless Sensor Network", 2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 141-148, 2018.