Comparison Functionalities of HTTP and MQTT Protocols

Tomislav Kušević(✉), Damir Blažević, and Tomislav Keser

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek, Osijek 31000, Republic of Croatia
tkusevic@gmail.com

Abstract. The mission and goal of this paper was to compare functionalities for both MQTT and HTTP protocol on the same example to see the advantages and disadvantages of sending and receiving data from server/device. Used tools are Android Studio, Kotlin, NestJS, TypeScript, PostgreSQL and in implementation two protocols were used: HTTP and MQTT protocols to determine advantages and disadvantages of the protocols, several examples are given where a particular protocol could be used in the development of mobile/IoT applications. While talking about MQTT and HTTP protocols, HTTP is one of the globally most used protocols for communication on the World Wide Web (Internet), when MQTT is for sure most used for connecting more devices in the networking called IoT (Internet of Things). Main goal is to show that MQTT can be also used in the mobile application, for example as notification service (server is notifying mobile devices).

Keywords: MQTT · HTTP · Android · Development · IoT · Protocols

1 Introduction

A few years ago, it was noticed an increasing number of devices that require connection to the Internet and access to Internet data. Main functionality is to connect multiple devices into one unit. There is also an increasing number of Internet protocols (HTTP, MQTT, etc.) that are used to connect these devices into one unit and manipulate data between them.

The main goal of the paper is to compare the functionality of the HTTP and MQTT protocols. It can be said that both protocols use the TCP/IP connection underneath, due to these similarities it is possible to compare them and get a clearer overview of the purpose of each protocol.

To successfully compare those two protocols, the paper will be based on practical work that will give us a more detailed overview of each of the protocols and how they work. Furthermore, this paper will be divided into two parts—the first part will be the practical part, previously described, while the second part is the theoretical part in which each protocol, its purpose and application will be described.

D. Blažević et al. (Eds.): OTO 2022, LNNS 592, pp. 45–56, 2023.
https://doi.org/10.1007/978-3-031-21429-5_5

1.1 Aim of the Paper

The goal of this paper is to create an application that will use the HTTP and MQTT protocol to connect and communicate with a remote server or other devices. This approach will help to have a clearer insight into each protocol, advantages, and disadvantages for both, based on which we will discern the purpose and expediency of the mentioned protocols. The application itself is quite simple, which gives us space and time to deal with the protocols themselves and their functionalities.

This application can add, edit, delete and read notes from a remote server. In addition, all devices will be notified when a change occurs in the database, so it will simulate the operation of an IoT system that mostly uses the MQTT protocol and try to find its usage in the development of mobile applications.

2 Used Technologies

In this part of the paper, the technologies that we used will be described will allow us to gain insight into the purpose and major features of the HTTP and MQTT protocols. It is well known that both HTTP and MQTT protocols are on the application layer of the TCP/IP protocol. To understand what the application layer is and what other layers are there, in the next chapter it will be explained what the TCP/IP protocol is and how it works.

2.1 General About the TCP/IP Protocol

TCP Protocol

TCP (Transmission Control Protocol) as a protocol on the application layer is responsible for creating the communication channels that make the network. It also determines how the message will be split into packets before they are sent to their destination using the network. Once the packets have successfully reached their destination, then the next step is to assemble the packets in the correct order so that they are understood and validated.

IP Protocol

IP (Internet Protocol) is a network protocol whose purpose is to route and address each packet so that each of them arrives at its destination. It is based on IP addresses written in packet headers. The most important data recorded in the IP structure are the IP address of the sender and the IP address of the destination [1].

TCP/IP Protocol

Using TCP and IP, TCP/IP (Transmission Control Protocol/Internet Protocol) was created. The first version that was more widely used was IPv4, which is still the dominant protocol used as communication over the Internet, later (in 2006) IPv6 was created—a more advanced version of the protocol that is gaining momentum. So, TCP/IP has been the standard for many years that provides computers and devices with remote communication over the Internet using packets [2, 3].

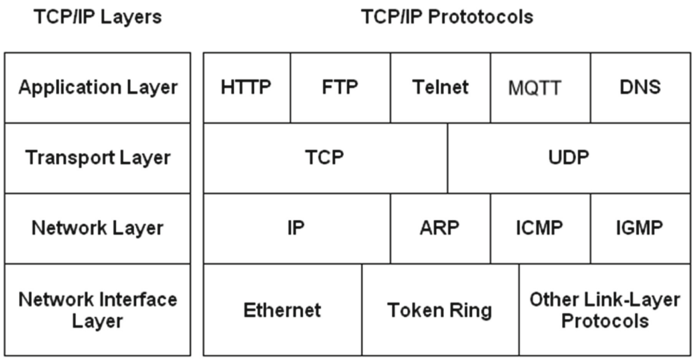


Fig. 1. TCP/IP architecture layers.

Each message is also sent in smaller packages that are usually sent by different routes to the same destination, where they are assembled and allow recipient to read the message at the destination. To understand how all of it is possible, it is best to describe each of them individually. TCP/IP consists of 4 layers: application, transport, network, and network access layers, which are shown in more detail in Fig. 1.

2.2 HTTP Protocol

The HTTP (Hypertext Transfer Protocol) protocol is an application protocol, which is the basis of the World Wide Web (Internet) data communication and is one of the most widely used protocols in the world. Currently, the HTTP/1.1 and HTTP/2.0 protocol versions are in widespread use (there is also HTTP/0.9). HTTP protocol communication works according to the request-response principle within the client-server architecture (Fig. 2). The most common clients are Internet browsers, mobile applications, client applications [5, 6].

Communication using the HTTP protocol takes place so that the client sends an HTTP request for a resource (e.g., an HTML page or a search operation) and receives an HTTP response from the server.

The request contains information about the identification of the resource (URL of the resource), the version of the protocol (e.g., HTTP/1.1), the HTTP method (GET, POST, PUT, DELETE, HEAD, OPTIONS or TRACE), request headers that define communication parameters (e.g., "Accept Language, Server: Apache 1.1") and request body with data to be sent to the server (e.g. user data in JSON format).

The response contains information about the protocol version, a status code (e.g., 200 OK) that provides the requesting client with information about the request processing process, response headers (e.g., Content-Type: text/html) and the response body with data that the server returns to the client (e.g., the corresponding JSON response).

Status Codes

- 100—Information while processing the request is still in progress
- 200/201—Success

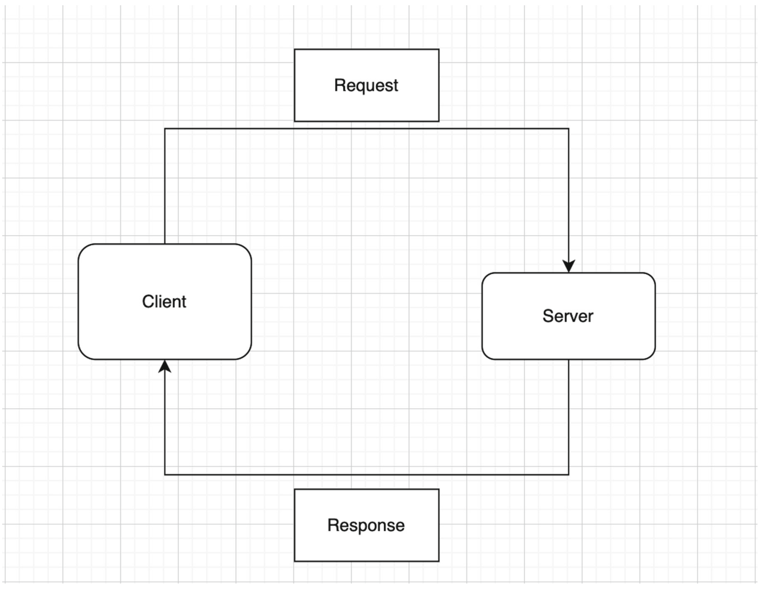


Fig. 2. HTTP Request-Response flow.

- 300—Redirect
- 400—Client error
- 500—Server error.

2.3 MQTT Protocol

General about the MQTT protocol:

MQTT (English Message Queuing Telemetry Transport) is one of the oldest M2M (English Machine to Machine) communication protocols. Andy Stanford-Clark from "IBM" and Arlen Nipper from "Arcom Control Systems Ltd" developed MQTT in 1999 [7, 8]. The main characteristic of the MQTT protocol is the lightweight messages, which are extremely suitable for IoT systems. MQTT consists of a client and an intermediary that communicate with each other so that the client publishes messages, while the server forwards the same messages to clients that have subscribed to that topic. Each message is posted to an address known as a topic. Clients can subscribe to multiple topics and receive every message published for each topic [7–10].

The most essential functions of the MQTT protocol:

Most often, when we talk about the MQTT protocol, we mention some of the basic functions that are necessary for the client (in this case, the device) to successfully receive data from the source (in this case, the broker), namely: connect, disconnect, publish, subscribe, unsubscribe [11].

2.4 Kotlin

Kotlin is a cross-platform programming language that is concise, secure, interoperable, and quite easy to use with other tools. It is a statically typed programming language that runs on the Java virtual machine and can also be translated into JavaScript source code, which makes it comprehensible to many developers who develop applications in programming languages: Java, JavaScript or TypeScript. Developed by the company "JetBrains", known for the development of powerful integrated development interfaces for the Java programming language called "IntelliJ IDEA". Today, Kotlin is the preferred programming language for developing Android applications—as confirmed by Google [12].

2.5 NestJS

NestJS is a platform (hereafter English framework) based on the Node.js platform, which is intended for use with the TypeScript (JavaScript) language, which aims to develop scalable applications on the server side as quickly and easily as possible. Applications developed on the NestJS platform are based on communication packages such as Express or Fastify. Nest is a new Node.js platform that not only imitates but also corrects the shortcomings of previous Node.js versions. When you start a new NodeJS project, NestJS is a much better choice than ExpressJS, because the intended project architecture is already defined in advance with a few simple components (controllers, modules, and services). This makes splitting applications into microservices simpler [13].

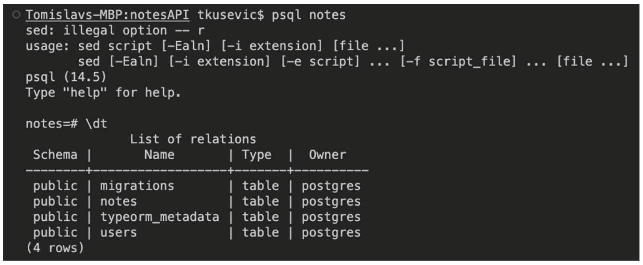
2.6 PostgreSQL

PostgreSQL is an open-source database. It is an object-relational database management system that stores data in rows, with columns as different data attributes. It allows you to store, process, and retrieve data safely. It was developed by a worldwide team of volunteers.

According to the DB-Engines PostgreSQL is currently ranked 4th in popularity amongst hundreds of databases worldwide [14]. PostgreSQL has built a formidable reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the commitment of the open-source community behind the software to consistently deliver efficient and innovative solutions. PostgreSQL runs on all major operating systems. It is no surprise that PostgreSQL has become the choice of many because of its features. We used PostgreSQL as a database where we store notes and users. Below is a list of tables within the database used in the project (Fig. 3).

Core features:

- PostgreSQL is compatible with Linux, Windows and macOS operating systems.
- PostgreSQL is highly secure, robust, and reliable. PostgreSQL supports multiple programming interfaces such as Java, C, C++, and Python.
- PostgreSQL supports various data types such as integer, string, and boolean. It also supports structured data types such as date/time, array, range. It can also work with documents like JSON and XML.
- PostgreSQL supports Multiversion Concurrency Control (MVCC) [15].



```
○ Tomislavs-MBP:notesAPI tkusevic$ psql notes
  sed: illegal option -- r
  usage: sed script [-Ealn] [-i extension] [file ...]
         sed [-Ealn] [-i extension] [-e script] ... [-f script_file] ... [file ...]
  psql (14.5)
  Type "help" for help.

  notes=# \dt
                List of relations
   Schema |       Name        | Type  |  Owner
  --------+-------------------+-------+----------
   public | migrations        | table | postgres
   public | notes             | table | postgres
   public | typeorm_metadata  | table | postgres
   public | users             | table | postgres
  (4 rows)
```

Fig. 3. PostgreSQL list of tables used in project.

3 Implementation of the Program Solution

The implementation of the software solution can be divided into 2 basic parts:

– Flow chart development
– Backend development
– Development of a mobile (Android) application that communicates with the server via HTTP protocol (Request & Response) and MQTT protocol (Publish & Subscribe).

3.1 Flow Chart Development

In the diagram (Fig. 4), clients (Android applications) connect to the server via the HTTP protocol with requests and wait for responses because of each request, and according to the MQTT Broker, after connecting, they have the option of pre-subscribing to a specific topic. Also, the server could connect to the MQTT Broker and with the "Publish" function, it could send a specific message to all clients.

3.2 Backend Development

For our application to function, it was necessary to initially create and develop prerequisites to save certain data in the database, clients (devices) have access to this data and can change, view, add and delete it. We used the previously mentioned PostgreSQL database as a database (local, later uploaded database on the server via the Heroku service). The access to the application itself is at the internet location: https://notes-mqtt.herokuapp.com/, and the documentation (for which we used an extension called Swagger) is at the internet location: https://notes-mqtt.herokuapp.com/api.

Data access is enabled on different routes and using different HTTP methods. All routes that can access the application are in the documentation and shown in the next picture (Fig. 5).

There are some of the most used routes that allow clients (devices) to access data from the server:

• POST/user/register—route used for user registration
• POST/auth/login—route used for user login

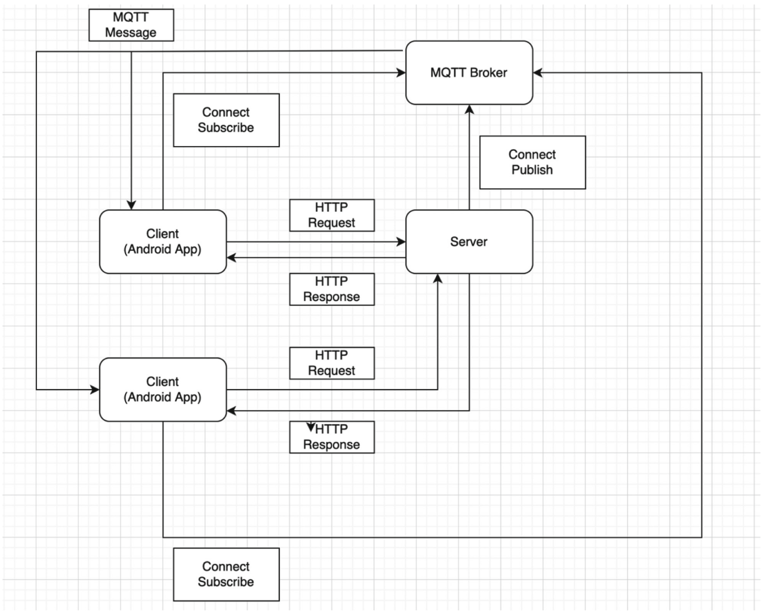
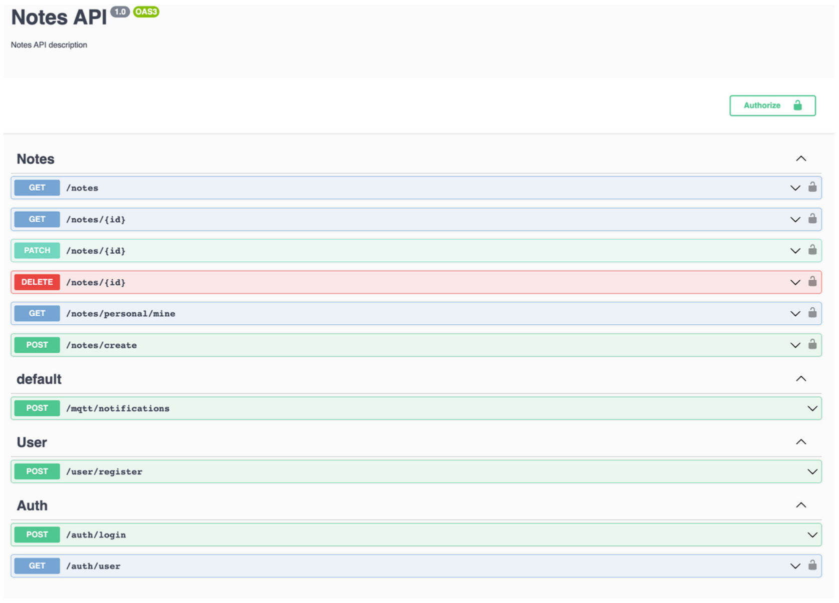


Fig. 4. Flow chart for the application solution.


Notes API 1.0 OAS3

Notes API description

Authorize

Notes

GET /notes

GET /notes/{id}

PATCH /notes/{id}

DELETE /notes/{id}

GET /notes/personal/mine

POST /notes/create

default

POST /mqtt/notifications

User

POST /user/register

Auth

POST /auth/login

GET /auth/user

Fig. 5. Swagger routes documentation.

- GET/auth/user—retrieving data of the currently logged in user
- GET/notes—get all notes

- GET/notes/:id—retrieving a specific note by identifier (id)
- DELETE/notes/:id—deletion of a specific note by identification mark (id)
- PATCH/notes/:id—edit a specific note by identifier (id).

3.3 Mobile (Android) Application Development

Functionality of the mobile application.

The developed Android application consists of:

- Splash screen
- Register screen
- Login screen
- Notes details screen
- Notes screen.

Splash screen

The functionality of the mobile application starts from the first initial window—the launch window (splash screen), where the memory in the device is checked to determine whether there is a logged-in user on the mobile device. If the user has already registered and logged in before, his identification number will be written into the memory, and the application will redirect him directly to the application, if the user has not previously logged in (or has deleted the application's internal memory), he will be redirected to the registration window.

Register screen

If the user does not have a registered account, this process is enabled on the register screen. The required registration parameters are e-mail and password. Besides the e-mail must be a valid e-mail, the password must consist of at least 6 characters to be valid. For registration, the server uses the JWT (JSON Web Token) package and the JWT strategy that enables the authentication process (registration and login). Also, after successful registration, the user will receive a token in the server's response with which he can access the rest of the call to the server (route).

Within each call, the client must send a token to get a valid response from the server, but also the server knows exactly which client wants access to the data and whether it has the right to the particular action it is trying to perform. The e-mail and password are saved in the database so that they can later validate a specific user during the login to the system.

Login screen

If the user has created an account within the application, then he can complete the application process on the application window. The registration process requires the e-mail and password of the previously created account in the registration window. If the login process is carried out successfully, the user is granted access to the application, more precisely to the notes window (list of notes).

Notes details screen

In addition to the list of notes, the user has the option of viewing all the details of a particular note. By clicking on a specific note, a new window will open showing all the details. When displaying the details, user see some of the following information: title of the note, text of the note, date the note was added, name of the author of the note. The user has the option to delete the note (if he is the author of it) by pressing the text "delete" in the upper right corner. In addition to deletion, the user also has the option to change the title and text of the note.

Notes screen

The main part of the application is the notes screen. Inside that window there are two tabs: the "All notes" tab and the "My notes" tab. In the notes screen there is a list of all notes in which user can see notes from all the users (Fig. 6 left), while in the "My notes" tab the user has an overview of the notes that he entered into the application (Fig. 6 right). Within the notes window, first the notes are retrieved from the server with an HTTP call and an MQTT connection is registered. The application is subscribed to a topic named "notes" which listens to all messages sent to the topic "notes"—this allows the user to retrieve data from the server again upon certain changes in the database (added or deleted notes) and to the list is refreshed.

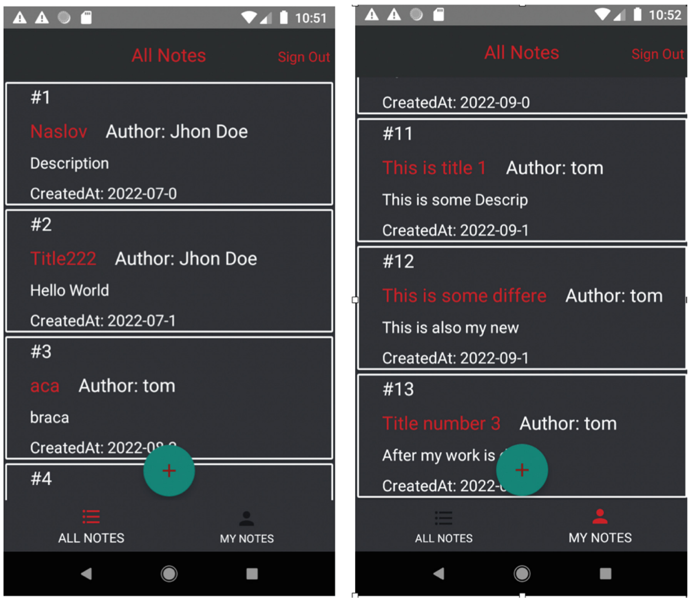
Thus, user have the possibility of refreshing the list of data without the need for manual refresh. So there user can see the advantages of using the MQTT protocol within a mobile application. The user can delete his own notes directly from the list of notes. It is also possible to add a note by pressing the "+" sign in the lower part of the screen, after which a window appears in which you can add the title and text of the note.

4 Comparison of MQTT and HTTP Used in Project

With the development of this application, MQTT has its future in the development of mobile applications. As it was already stated, the HTTP protocol has been used for many years and in an exceptionally significant percentage for retrieving data from the Internet (servers). Although the MQTT protocol is mostly used in IoT systems, now they have an example of using this form of protocol in the world of mobile applications as well [16]. The real advantage of MQTT over HTTP occurs when we reuse the single connection for sending multiple messages in which the average response per message converges to around 40 ms.

When choosing MQTT over HTTP, it is so important to reuse the same connection as much as possible. If connections are set up and torn down frequently just to send individual messages, the efficiency gains are not significant compared to HTTP.

Greatest efficiency gains can be achieved through MQTT's increase in information density for each payload message. Most straightforward approach is to reduce the payload size where more data can be transmitted in each payload, which can be achieved through choosing proper compression and package methods based on the type of the data being generated [17]. For an easier comparison, follow three tables can show certain information and data about the use of both protocols (Tables 1 and 2).


10:51
All Notes Sign Out
#1
Naslov Author: Jhon Doe
Description
CreatedAt: 2022-07-0
#2
Title222 Author: Jhon Doe
Hello World
CreatedAt: 2022-07-1
#3
aca Author: tom
braca
CreatedAt: 2022-08-2
#4
ALL NOTES MY NOTES

10:52
All Notes Sign Out
CreatedAt: 2022-09-0
#11
This is title 1 Author: tom
This is some Descrip
CreatedAt: 2022-09-1
#12
This is some differe Author: tom
This is also my new
CreatedAt: 2022-09-1
#13
Title number 3 Author: tom
After my work is
CreatedAt: 2022-
ALL NOTES MY NOTES

Fig. 6. Display of two tabs in Notes screen.

Table 1. Specification table for HTTP and MQTT.

	MQTT	HTTP
Name	MQ telemetry transport	Hyper text transfer protocol
Arhitecture	Publish/subscribe	Request/response
Aim	Topic	URL
Based on the protocol	TCP/IP	TCP/IP
Security	TLS + username/password	TLS + username/password
Type of connection	Connection status known	Connection status not known
Sending process	Async, event based	Sync
Queue	Broker can remember requests and informations	Based on implementation
Header size	2 bytes (binary)	min. 8 byte (text)
Message size	max. 256 MB	–
Message type	No type specified (mostly binary)	Text (Binary-Base64)
Message distribution	One-to-Many	One-to-One
Reliability	0—fire and forget 1—at least once 2—at most once	Based on implementation

Table 2. Comparison of memory usage in HTTP and MQTT functions.

Function name	MQTT (B)	HTTP (B)
Connect	5572	2261
Disconnect	376 (optional)	0
Sending message	388	3285
For 1 message	6336	5546
For 10 messages	9829	55046
For 100 messages	44,748	554,600

Table 3. Response time comparison for HTTP and MQTT.

No. of messages	MQTT—response time (ms) (QoS 1)	HTTP response time (ms)
1	113	289
100	47	289
1000	43	289

5 Conclusion

By comparing the HTTP and MQTT protocols on a simple application, there are advantages, disadvantages and intended operation of the protocol [15–17]. HTTP protocol is mostly used when a large amount of data should be sent (e.g. large JSON) to save certain data in the database. Every time client open an HTTP call, receive data, the connection is closed.On the other hand, the MQTT protocol is an excellent "light" protocol that allows us to connect two devices (subscribe to a topic) at once, and their communication is possible without closing it. If a device is subscribed to the topic, it will receive messages continuously.

In addition to two-way communication (a device can receive messages on a topic, but also send messages on the same topic), MQTT also allows us to communicate with multiple devices. Let us say there are 2 devices in the house that constantly send changed data (e.g., thermostat) that measure temperature and humidity—such a device can send data to a specific topic to all other devices that are connected to it. With that alone, it is necessary to send a message to the topic once, and all other subscribed devices will receive that message (the so-called broadcasting Data).

MQTT can also be used as a protocol for sending notifications to devices: Consider that there is an application that deals with writing and editing documents/notes. If two or more people were to edit the same document, a race-condition could occur over the data where one user would save the document after changes, and the other user would not catch those changes because he opened the document before editing and overrode the previous user's changes—in that situation MQTT can be of help to us. Each user, when opening the document, would subscribe to the topic, in case of a change to the document—all users would receive a notification that the document has been changed and to withdraw the last changes. Thus, race-condition is not occurring anymore. There is also some greater implementation of the MQTT protocol in the IoT industry (device communication), but also in application development—some form of the notifications.

References

1. What is the Internet Protocol? https://www.cloudflare.com/learning/network-layer/internet-protocol/. Last accessed 24 Sept 2022
2. Tobey, P.: IPv4 and Education: The Basics (2022). https://circleid.com/posts/20220607-ipv4-and-education-the-basics. Last accessed 24 Sept 2022
3. Ames, I.: TCP/IP Protocol Suite (2019). https://medium.com/software-engineering-roundup/tcp-ip-protocol-suite-ec7ed4888d5d. Last accessed 24 Sept 2022
4. GeeksForGeeks: Transport Layer Responsibilities. https://www.geeksforgeeks.org/transport-layer-responsibilities/. Last accessed 24 Sept 2022
5. Grigorik, I.: High Performance Browser Networking. O'Reilly Media (2013)
6. Fielding, R., Reschke, J.: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, RFC 7230. Internet Engineering Task Force (IETF) (2014)
7. Egli, P.: MQTT—Message Queueing Telemetry Transport Introduction to MQTT, A Protocol for M2M and IoT Applications (2017)
8. Šušnja, M.: Primjena mqtt protokola u internetu stvari, Specijalistički diplomski stručni, Sveučilište u Splitu, Sveučilišni odjel za stručne studije. Split (2020). https://urn.nsk.hr/urn:nbn:hr:228:047974
9. Bensakhria, A.: IoT—Communicating with Devices: Introduction to MQTT and HTTP (2020)
10. MQTT vs. HTTP: Which One Is the Best for IoT? https://medium.com/mqtt-buddy/mqtt-vs-http-which-one-is-the-best-for-iot-c868169b3105. Last accessed 24 Sept 2022
11. MQTT Beginner's Guide? https://www.u-blox.com/en/blogs/insights/mqtt-beginners-guide. Last accessed 24 Sept 2022
12. Heller, M.: What is Kotlin? The Java Alternative Explained. https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html. Last accessed 23 Sept 2022
13. An Introduction to MQTT with NestJS Framework. https://myas92.medium.com/an-introduction-to-mqtt-with-nestjs-framework-a69ec55483f0. Last accessed 24 Sept 2022
14. PostgreSQL Tutorial. https://www.simplilearn.com/tutorials/sql-tutorial/postgresql-tutorial. Last accessed 25 Sept 2022
15. What is PostgreSQL and Why Do Enterprise Developers and Start-ups Love It? https://canonical.com/blog/what-is-postgresql. Last accessed 25 Sept 2022
16. MQTT Vs. HTTP for IoT. https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/. Last accessed 24 Sept 2022
17. HTTP vs. MQTT: A Tale of Two IoT Protocols. https://cloud.google.com/blog/products/iot-devices/http-vs-mqtt-a-tale-of-two-iot-protocols. Last accessed 25 Sept 2022