A messaging application cannot function without instant messaging protocols that execute the real-time communication between users. Knowing how they work is vital to understanding what's possible when it comes to building a chat application.

The most popular protocol on the market is XMPP. That doesn't mean, however, that Matrix and MQTT don't have anything to offer. Each protocol has its pros and cons and can be useful for different purposes, so, as in case of most software development projects, everything comes down to understanding your needs and priorities.

I sat down with Paweł Dorofiejczyk , our Solution Architect and expert in communications technology, to help you make the right decision as you pick the messaging protocol for your chat application.

What is XMPP?

XMPP stands for Extensible Messaging and Presence Protocol (it was formerly known as Jabber). It is an open source technology that is expandable and supports the transmission of current information, including data. WhatsApp, the most popular messaging app in the world, is built with XMPP. For other examples I suggest you take a look at 21 more companies that use XMPP in their applications .

XMPP follows the client-server architecture pattern. To apply XMPP effectively, you have to pass it through an appropriate transport binding such as TCP/IP or HTTP.
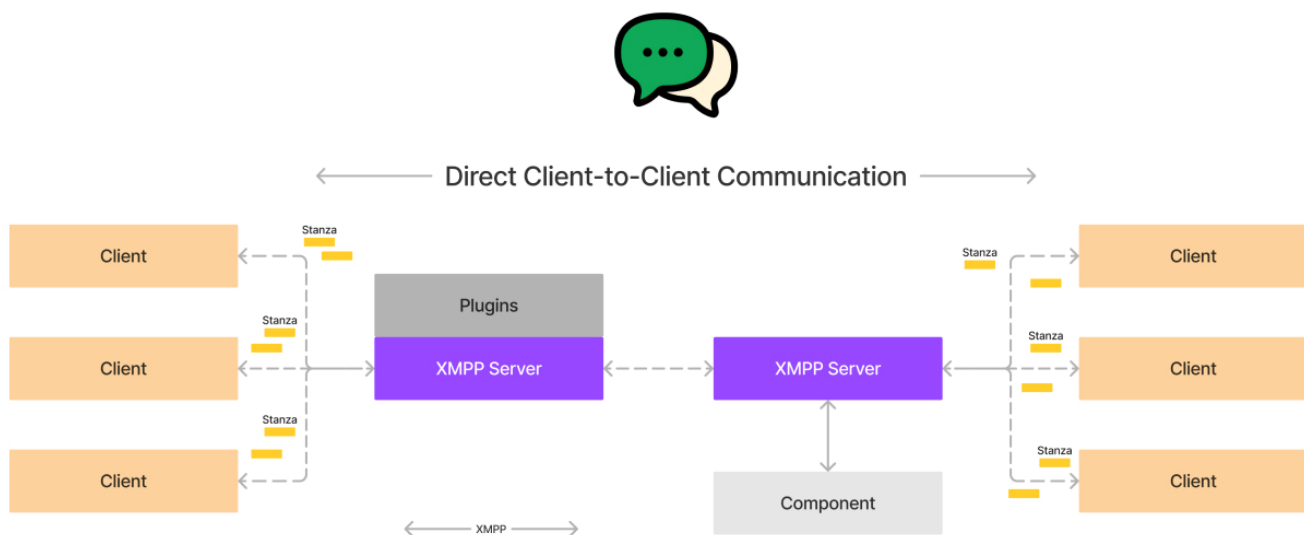
Pros of XMPP

Decentralized architecture

The protocol uses a client-server model which means that users do not interact with each other directly; there is no central server (like in e.g. Telegram) that relays messages. Instead, every provider on the XMPP network runs their own server and every user is allocated an XMPP address (JabberID) that works like an email address with an IP address/domain name and a username for the resident server.

Top-notch security built into the presence protocol

Users are required to authenticate both host servers and messages to avoid the risk of spoofing and reduce the risk of spam. Personal identity in the XMPP protocol is strong. Additional security layers can be implemented by requiring clients to install valid security certificates for identity confirmation.

Multiple Extension Protocols

XMPP offers a lot of Extension Protocols (or XEPs) that define concrete functionalities of the chat application, making it a lot easier for developers to build chat applications.

Two types of encryption

With XMPP, the first encryption takes place in the application using SASL. Once a connection is established, all client-server exchanges are encrypted with Transport Layer Security (TLS). Such double encryption significantly lowers the risk of getting attacked.

Effective support community

The extensive support community is one of the greatest advantages of XMPP. It can be easily contacted via the protocol's official website .
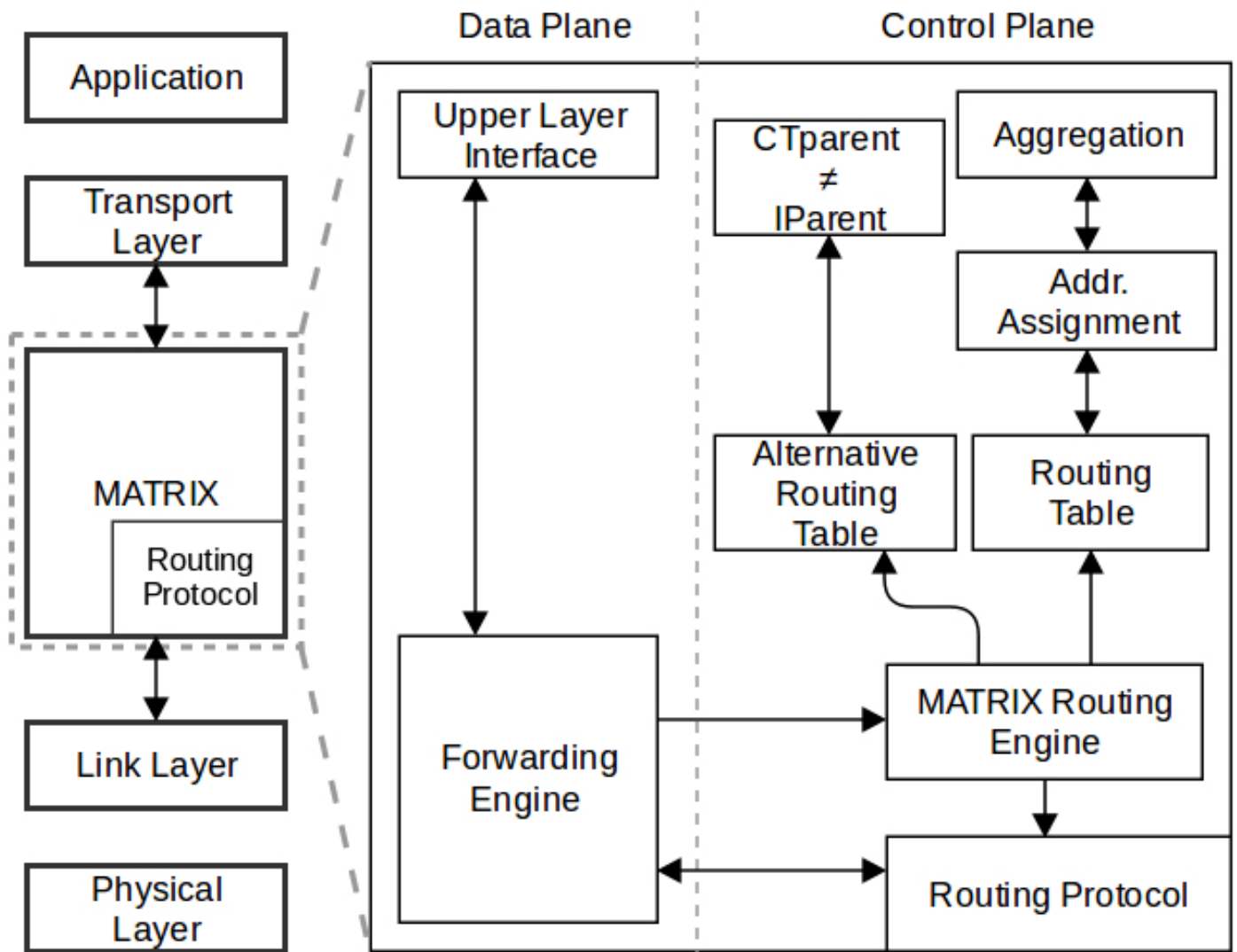
Cons of XMPP

- XMPP is relatively slow.
- The XMPP system isn't very effective when it comes to sending binary data.

What is Matrix?

Matrix is another chat protocol you could consider for your chat application. It's an open standard for interoperable, decentralized, real-time communication over IP. It's growing in popularity as an alternative to XMPP. It was recently added to Ejabberd.

Matrix can be used to power instant messaging, IoT communication, VoIP/WebRTC signaling – anything that requires a standard HTTP API for publishing and subscribing to data while tracking conversation history.

In comparison to XMPP, Matrix will be a better choice for applications like Slack – those that support communication around organized groups or topical conversations.

Data Plane **Control Plane**

Application

Transport Layer

Upper Layer Interface

CTparent ≠ IParent

Aggregation

MATRIX

Routing Protocol

Addr. Assignment

Alternative Routing Table

Routing Table

Link Layer

Forwarding Engine

MATRIX Routing Engine

Physical Layer

Routing Protocol
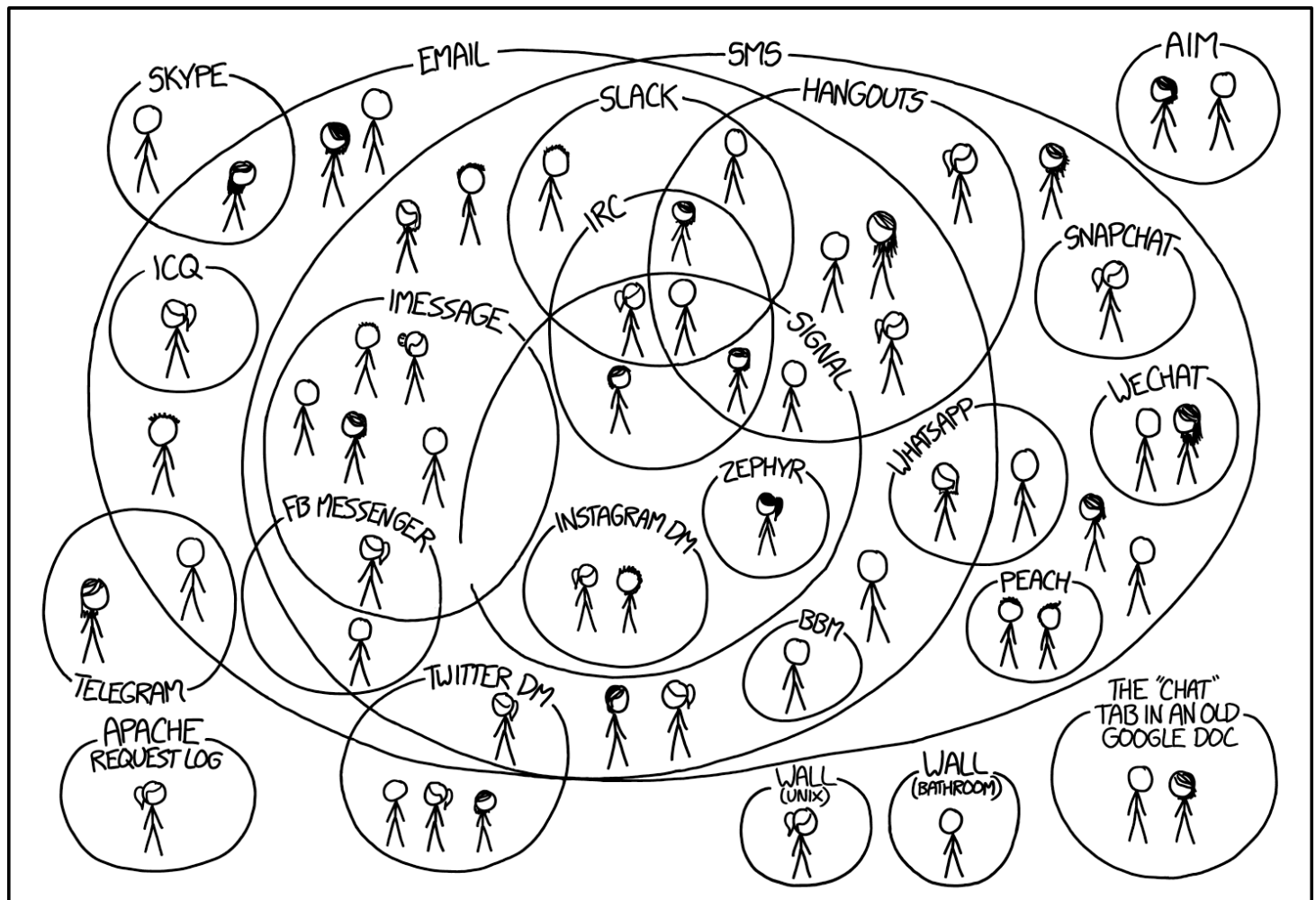
Pros of Matrix

End-to-end encryption

The Matrix server encrypts all chat and private room conversations with end-to-end encryption by default. This is a convenient feature that isn't available in most XMPP servers.

Intuitive by default

Users would typically expect a message they receive to show up on all of their devices, not just the one that picks it up first. They also expect to be able to view the chat history when they install an application on a new device. The default settings in Matrix support exactly that: chat conversations powered by Matrix work like entire chat histories that can be edited and synced by multiple people. That said, XMPP also allows you to achieve this, but it requires greater configuration efforts.
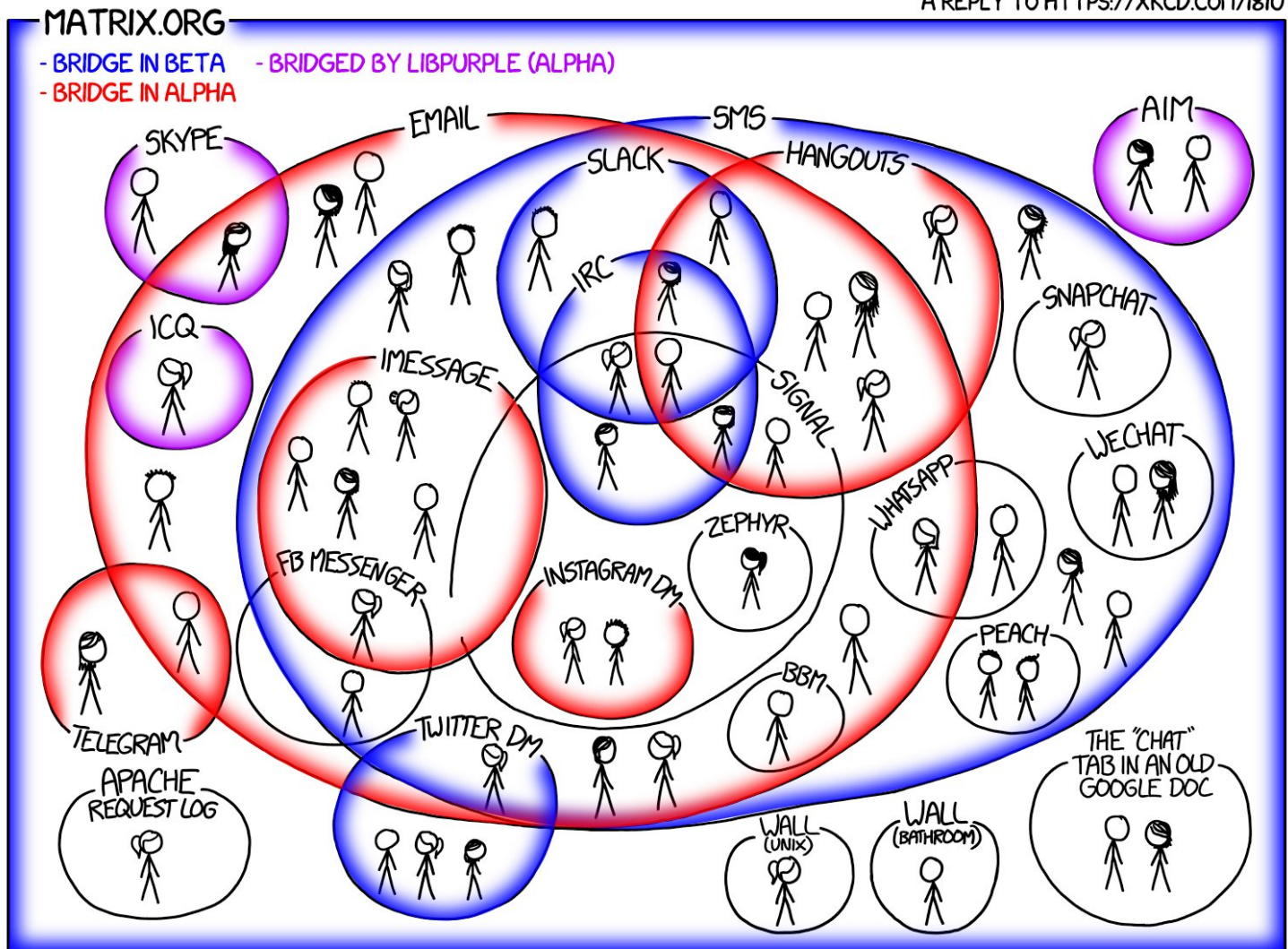
Decentralized architecture

Matrix works more like an email service, in that the messages can be intercepted by various clients, regardless of the application they are using. This basically eliminates the single point of control or failure.

I HAVE A HARD TIME KEEPING TRACK OF WHICH CONTACTS USE WHICH CHAT SYSTEMS.

Source: https://xkcd.com/1810/

Source: https://twitter.com/matrixdotorg/status/841424770025545730/photo/1

Cons of Matrix

- The rooms in Matrix are replicated over all participating servers – there is no focal point as there is in XMPP multi-user chat.
- Configuration options offered by Matrix are rather limited, especially in comparison to XMPP which is supported with a good command-line.
- With Matrix, the conversation metadata is known to all servers (this includes information on the accounts that are involved and those that were made public).

What is MQTT?

MQTT stands for MQ Telemetry Transport. It was formerly known as Message Queuing Telemetry Transport. It is a lightweight protocol that uses minimum CPU resources, so it's best for building small, low-power chat

apps.

It's a well-established technology created in 1999 for machine-to-machine communication that works well in challenging environments, such as high latency, low bandwidth and unreliable networks. It is generally used in embedded environments.

The MQTT messaging architecture is based on a publish/subscribe pattern. Consequently, publishers and subscribers are decoupled, which means they don't have to establish a direct connection, nor need to be present online at the same time.

MQTT is used by Facebook in its mobile chat application (more on this on Engineering at Meta), as well as Instagram in its direct message functionality.

How MQTT works
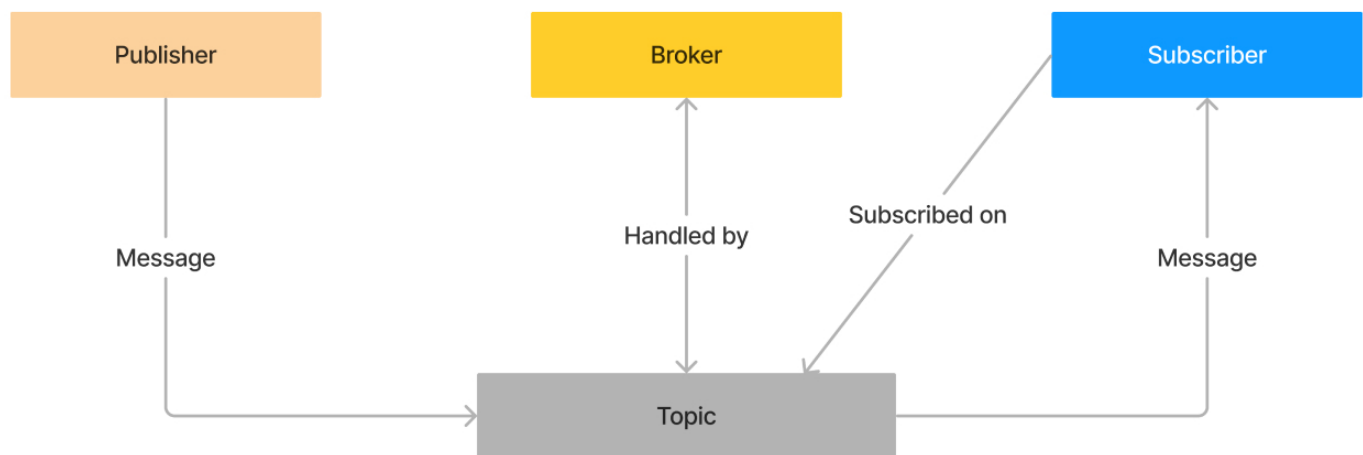
Pros of MQTT

Reliability

MQTT can transfer data even with unstable connections. It offers three Quality of Service (QoS) levels to support that.
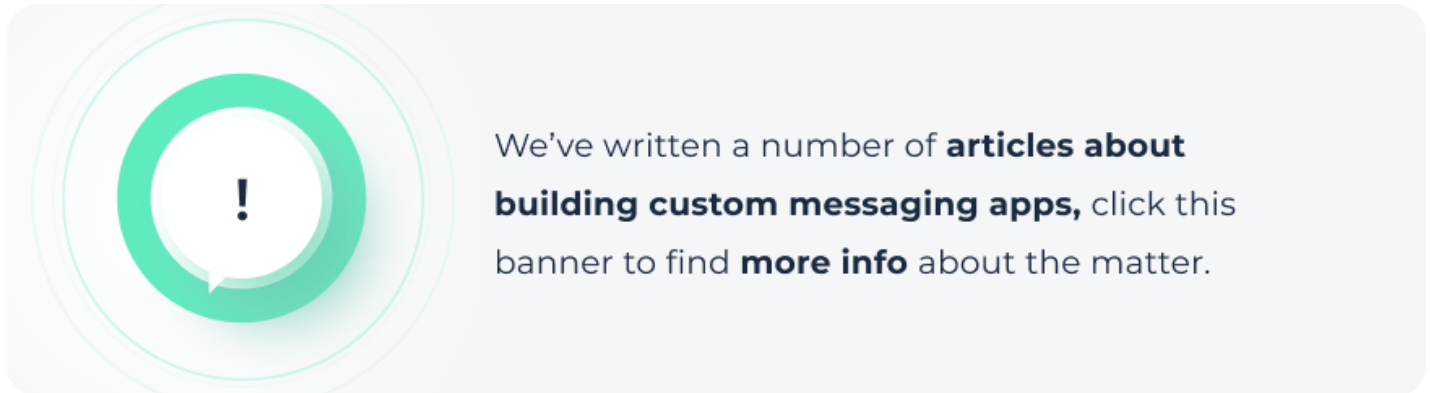
Open and easy to use

A well-defined, standard documentation is one of MQTT's greatest assets. It offers a set of ready-made libraries for Java, Python, C++, JavaScript and other programming languages.

Cost-effective

The protocol is cheap and super easy to implement, thanks to the free available documentation.

Cons of MQTT

- MQTT relies on a centralized broker, which may limit scalability.
- Doesn't offer as many ready-to-use solutions for live chat building as XMPP. It provides the basic layer for sending messages or events (which can also be an advantage, depending on your goals).

We've written a number of **articles about building custom messaging apps,** click this banner to find **more info** about the matter.

Comparing XMPP vs Matrix vs MQTT instant messaging protocols for chat app development projects

To pick the best chat protocol for your real-time messaging app, you need to start by reviewing your business objectives. If you want to benefit from the client-server architecture pattern or build an app like WhatsApp, opt for XMPP, as long as you won't focus heavily on binary data.

If you are about to create an app like Slack or a system that will be powering group communication, you can benefit from Matrix's decentralized architecture that synchronizes conversation history within a room, not message passing.

If you want to implement your app fast to support machine-to-machine communication in challenging environments, MQTT will serve you well. It allows transmitting data effectively and is a good choice for networks that experience different levels of latency.

If you're still unsure about which protocol to select for your next chat application, contact me directly at   magda@rst.software   and I'll put you in touch with our chat application development experts who will help you make the best decision.