Communication based on MQTT protocol

Milan Milojić[1]
[1]*Morena Inženjering, Niš*

Abstract – *MQTT is a messaging protocol that enables efficient and lightweight telemetry transport that is required in the modern era of communications. Many devices are communicating directly to each other, thus forming the Internet of things, a new Machine to Machine (M2M) communication network. Aim of this paper is to present MQTT key concepts, most notably publish/subscribe model that is well suited for smart sensors and smartphones, as they require efficient means of communication due to their limited processing power and short battery life. Some successful MQTT application examples will be considered and it will be discussed on how they can benefit today's world. It will be discussed with more detail how Android devices among others can benefit most out of MQTT, by using a Service implementation of a push mechanism for receiving important notifications.*

1. INTRODUCTION

Message Queue Telemetry Transport (MQTT) is a messaging protocol that is lightweight enough to be supported by the smallest devices, yet robust enough to ensure that important messages get to their destinations every time. With MQTT devices such as smart energy meters, cars, trains, satellite receivers, and personal healthcare devices can communicate with each other and with other systems or applications. [1]

There is a revolution happening right now, the whole client-server concept and the way we experience internet is rapidly changing. "Internet of things"[2] is here, "things" as diverse as smartphones, cars and household appliances to industrial-strength sensors are being linked to each other and the internet. And while we currently only have some simple intercommunication and autonomous machine-to-machine (M2M) data transfer, the potential benefits to lifestyles and businesses are huge.

Telemetry technology allows us to measure and monitor things from a distance. By improving technology we made it possible to interconnect devices at remote places and reduce cost of building and maintaining these systems. Information is more important today that any gold, black or regular, and challenges lie in getting the information from the devices to the people and applications they are using in timely manner. Information is only as good as the time received allows it to be and ability to respond accordingly only increases its value. If the devices are widely distributed geographically, or if they have limited storage or computational abilities, the challenges increase considerably, as do costs. Fortunately, these challenges are being overcome through the use of improved telemetry technologies and communication protocols that are making it possible to send and receive this information reliably over the Internet, even if the network is unsteady or the monitoring device has little processing power. MQTT provides telemetry technology to meet the information challenges of today's internet users.

2. MQTT PROTOCOL KEY CONCEPTS

As we said previously MQTT is very simple and lightweight messaging protocol with open and easy to implement server architecture which can support thousands of remote clients. As such it is ideal to be used in environments with low network bandwidth and/or high latency. Also it does not require powerful devices to run; instead it can be used on devices that have very little memory and processing power. Key concepts MQTT protocol is built upon are all aimed at overcoming aforementioned obstacles in the best possible way.

Publish/subscribe

MQTT protocol follows the Observer design pattern where one end is publishing messages while the other end is listening ("observing") for those messages. Clients are subscribed to *topics* of interest and receive any messages that are published to those topics.

Topics and subscriptions

Topics are subjects of interests for users. Messages in MQTT protocol are published to those topics and clients, in turn, signup to receive messages by subscribing to a topic. Subscriptions can be explicit, and received messages are

limited only to that topic, or can be more general by using wildcard designators and thus receiving messages for many related topics.

Quality of service levels
As with any network protocol, quality of service (QoS) plays an important role. MQTT defines three levels for message delivery, each level gradually increasing the level of effort by the server to ensure that the message gets delivered. Higher QoS comes at a price of increased bandwidth consumption and latency.

Retained messages
It is important that any new client that subscribes to a topic does not miss out on any previous messages. That is why with MQTT, the server keeps the messages even after they are sent to all of the subscribers. When a new client subscribes to an existing topic, any retained messages are being sent to the new subscribing client.

Clean sessions and durable connections
Optional clean session flag is used to mark whether client subscriptions are removed when it disconnects from the server. If the flag is set to true then all of the client's subscriptions are removed, otherwise the connection is treated as durable and the client's subscriptions remain in effect after any disconnection. This has the effect of keeping any messages that are received after disconnection and resending them after the connection is reestablished.

Wills
When a client connects to a server, it can inform the server that it has a will, or a message, that should be published to a specific topic or topics in the event of an unexpected disconnection. A will is particularly useful in alarm or security settings where system managers must know immediately when a remote sensor has lost contact with the network.

MQTT brokers
MQTT broker is a server that implements the MQTT protocol. It is used to relay messages from clients publishing messages to a certain topic to those who are subscribed to it. WebSphere MQ is the best known broker as it is developed and maintained by MQTT founders, IBM. Other open source broker implementations are Really Small Message Broker (RSMB) and Mosquitto. Mosquitto is an open source message broker that implements the MQ Telemetry Transport protocol version 3.1.[3]

3. APPLICATIONS

One of the first projects that popularized MQTT protocol was "Andy's Twittering House". Dr. Andy Stanford-Clark, one of the authors of MQTT gained media attention in the late 2000s by connecting his home automation system via MQTT to Twitter, a popular micro-blogging site.[4][5]

Of course, sensors tweeting about what is going on in your house may not be that much of a breakthrough, but the idea was to get people's attention. Any messaging protocol can work in two directions, so if we were to reverse the whole "tweet house" idea, we could tweet to our house to turn off the lights or turn on the air-conditioning. Even though this may not seem as a smart thing to do it is a good proof of concept. MQTT provides users with efficient transport protocol for sending commands to home automation system from a remote location. And because it is so lite, it doesn't require huge investments. MQTT broker such as Mosquitto can run even on 35$ Raspberry Pi.[6]

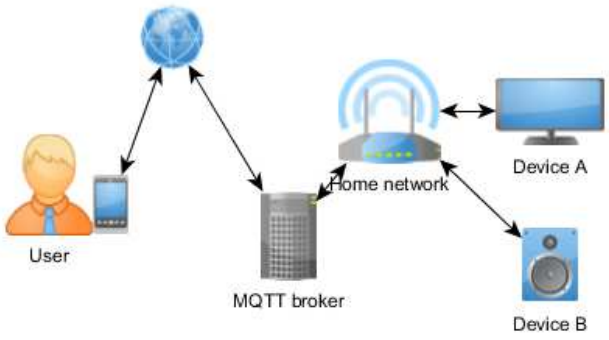

Figure 1: Home automation system controlled from smartphone

As shown in Figure 1, user can use his smartphone to send command over internet to MQTT broker which would publish that command. Home automation server that can control various devices inside of the home would be subscribed to that topic and upon receiving the message it would issue a command to a device targeted inside the message. And if the users subscribes to the predefined topic he can receive command response, statuses and notification from the sensors that are directly connected to those devices. Smoke detector in the kitchen, humidity sensor on the bathroom floor or window opening alert could trigger an alarm that would be published to MQTT broker. Home owner need not be the only person subscribed to these topics. Fire department, police and etc. can adapt their systems to receive these messages as well. This can provide

the user with the necessary information needed to deal proactively with everyday situations that arise with home automation.

Another popular project relying heavily on MQTT is “FloodNet”, initiated by the University of Southampton. Their primary goal is to demonstrate a methodology whereby a set of sensors monitoring the river and functional floodplain environment at a particular location are connected by wireless links to other nodes to provide an "intelligent" sensor network. [7]

Unlike many other systems FloodNet focuses on power conservation and low maintenance of instrumentation in the field and this is best achieved by using MQTT protocol. Figure 2 below describes the system architecture of FloodNet. The ad hoc network is based on 802.11, and consists of powerful nodes that host IBM's Websphere MQ software. The nodes which are essentially sensors transmit data at regular intervals via GPRS to a micro-broker or gateway, and the gateway subsequently transmits the data to an IBM's Messaging Broker. The sensor data is transcoded and transformed at this end, and is delivered to any application that subscribes to the data. The sensor data might be used for various applications such as simulation models, GIS Visualization and database services.

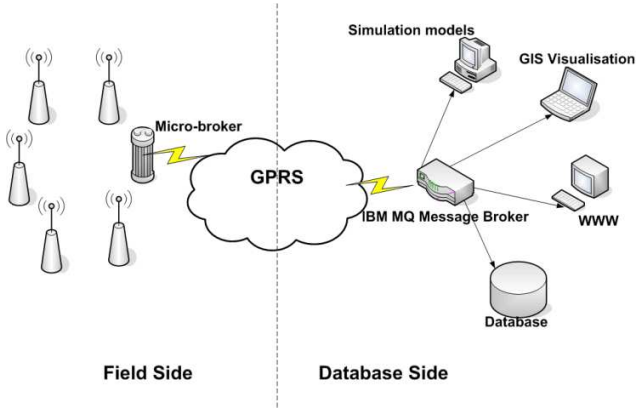

Figure 2: FloodNet architecture [8]

In healthcare MQTT has been successfully implemented in a solution for cardiac patient monitoring system. The solution needed to address the following aspects of patient care:

- Monitoring cardiac patients after they leave the hospital
- Improving the efficiency of later checkups
- Meeting new industry data-capture standards

The solution included an embedded MQTT client in a home monitoring appliance that collects diagnostics whenever the patient is in close proximity to a base unit. The base unit sends the diagnostic data over the Internet to the central messaging server, where it is handed off to an application that analyzes the readings and alerts the medical staff if there are signs the patient might be having difficulty.

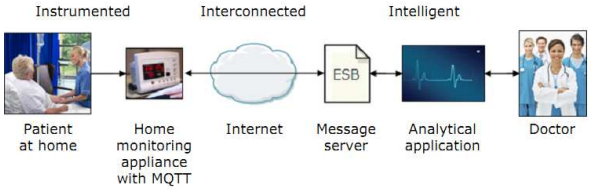

Figure 3: Home pacemaker monitoring solution with MQTT

This solution provides patients with better and more proactive level of service from their hospitals and increases their chances for recovery because they are constantly monitored by their appointed doctors. It can also save money as patients have less need to travel and schedule an appointment with their doctors.

4. ANDROID CHAT APPLICATION BASED ON MQTT PROTOCOL

Instant message communication is becoming more and more integrated into everyday mobile phone use as phones are getting smarter and data plans cheaper. SMS (Short message service) is rapidly becoming obsolete in modern times where “on-line” time is equal to 24 hours. Main advantage that SMS had was its instant delivery time. In practice, you were guaranteed that you would receive a SMS message within seconds, provided you had network coverage.

In order to achieve this on mobile clients running instant messaging services we need an efficient message push mechanism. This is a very important concept because many other applications, not just instant messaging applications, rely on efficiently responding to received data. A widely used approach is to poll data from the server periodically but this may result to delays. If an application polls too often the device’s battery will be drained too quickly due to frequent processor activation. If it pools infrequently then it might respond to changes slowly or even miss an important event. There is no good compromise here. The best solution here would be for an application to just wait and receive the message when it is sent, just as with SMS. Big companies like Google, Apple, BlackBerry, various mobile providers,

have their own closed protocols for implementing push messaging. And this is where MQTT protocol fits perfectly.

One other issue with mobile devices is the noticeable latency issue. When social networking giant Facebook created its first instant messaging application for mobile clients, latency between messages was multiple seconds. Large and scalable platform that has been used by millions of users was struggling with efficient message delivery as the method they used was reliable but slow and there were limitations for improvement. This was unacceptable and they needed a quick solution for this problem. They ended up building a new mechanism that maintains a persistent connection to our servers and to do this without killing device battery life MQTT protocol was chosen. By maintaining an MQTT connection and routing messages through our chat pipeline, they were able to often achieve phone-to-phone delivery in the hundreds of milliseconds, rather than multiple seconds.[9]

The easiest way to start using MQTT protocol on Android platform is by implementing Eclipse Paho project. "The Paho project has been created to provide scalable open-source implementations of open and standard messaging protocols aimed at new, existing, and emerging applications for Machine-to-Machine (M2M) and Internet of Things (IoT). Paho reflects the inherent physical and cost constraints of device connectivity. Objectives include effective levels of decoupling between devices and applications, designed to keep markets open and encourage the rapid growth of scalable Web and Enterprise middleware and applications. Paho initially started with MQTT publish/subscribe client implementations for use on embedded platforms, and in the future will bring corresponding server support as determined by the community". [10]

Under the assumption that there is already an instant messaging application developed for Android it will be demonstrated here how such application can benefit from push notification based on MQTT protocol. In order for an Android application to receive push notifications it needs to be active. Due to operating system limitations only one Android Activity can be active at any time. On the other hand there can be many active Services on Android. "A Service is an application component representing either an application's desire to perform a longer-running operation while not interacting with the user or to supply functionality for other applications to use".[11] However, it is important to realize that Service is not a separate process nor it has its own thread. While this may not be appropriate for use "out of the box", it is possible to create a separate worker thread when creating the Service object in *onCreate()* method. The Android system will attempt to keep the process hosting a service for as long as the service has been started or has clients bound to it. Process manager will award it the highest priority when the system is running low on memory and there are decisions to be made about process killing.

But this doesn't guarantee us that our connection to the broker will be consistent. One way for us to achieve persistent TCP/IP connection is to periodically ping broker just to keep the connection alive. To ensure that the connection wouldn't be lost when, for example, the device goes to sleep, it is possible for us to get the Wake Lock on the system in order to control the connection keep-alive process. WakeLock is part of the PowerManager class available since Android API level 1.[12] But as we mentioned before when we talked about poll mechanism, this isn't a good solution as it may waste device's battery.

A better solution is to AlarmManager class that is available in the Android API. This class provides access to the system alarm services. These allow you to schedule your application to be run at some point in the future. When an alarm goes off, the Intent that had been registered for it is broadcast by the system, automatically starting the target application if it is not already running. The Alarm Manager is intended for cases where you want to have your application code run at a specific time, even if your application is not currently running.[13] This allows us to schedule when the next server ping needs to happen i order for connection to stay alive.

By implementing an Android service in this way, it allows us to receive messages from the broker almost instantaneously. And because MQTT messaging protocol is lite it does not eat much bandwidth. Received message can then trigger various actions in the hosting application. In the case of instant messaging application, Android application would subscribe to a topic that is equal to user's unique name. When user receives a new message and he is not connected to the instant message communication server, that server would publish a message to user's topic, informing the user about pending messages. Android application would receive published MQTT message and then it could download pending messages from the communication server and notify user that there are new received messages.

5. CONCLUSION

Using the MQTT protocol extends our reach to tiny sensors and other remote telemetry devices that might otherwise be unable to communicate with a central system or that might be reached only through the use of expensive, dedicated networks. Network limitations can include limited bandwidth, high latency, volume restrictions, fragile

connections, or prohibitive costs. Device issues can include limited memory or processing capabilities, or restrictions on the use of third-party communication software. In addition, some devices are battery-powered, which puts additional restrictions on their use for telemetry messaging. By being simple and open, MQTT protocol helps us overcome these obstacles by relying on a publish/subscribe model.

There are many examples of successful implementations in healthcare where devices that communicated on MQTT protocol were used for monitoring cardiac patients and capturing potentially life-saving data. Utility companies that produce energy are using MQTT for monitoring remote locations and collecting important maintenance and failure data. But perhaps most interesting application that has the broadest reach is social networking. Social networking giant Facebook has been using MQTT for their chat service as it has proven to be reliable and much faster than their previous solution.

6. REFERENCES

[1] V. Lampkin, W. T. Leong, L. Olivera, S. Rawat, N. Subrahmanyam, R. Xiang, *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*, IBM Redbooks, September 2012

[2] Wikipedia, *"Internet of things"*, http://en.wikipedia.org/wiki/Internet_of_Things, retrieved 1. 2. 2013.

[3] http://mosquitto.org/, retrieved 1. 2. 2013.

[4] "Tweeting mouse trap and window", http://news.bbc.co.uk/2/hi/technology/8113914.stm, retrieved 1. 2. 2013.

[5] "Andy's house on Twitter", https://twitter.com/andy_house, retrieved 1. 2. 2013.

[6] Raspberry PI Adventures, http://rasspberrypi.wordpress.com/2012/09/16/mosquitto-mqtt-on-raspberry-pi-broker-publish-and-subscribe-client/, retrieved 1. 2. 2013.

[7] FloodNet Project, http://envisense.org/floodnet/floodnet.htm, retrieved 1. 2. 2013.

[8] FloodNet System Overview, http://envisense.org/floodnet/images/system_architecture.gif retrieved 1. 2. 2013.

[9] Facebook.com, *"Building Facebook Messenger"*, https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920, retrieved 1. 2. 2013.

[10] Paho Project, http://wiki.eclipse.org/Paho, retrieved 1. 2. 2013.

[11] Android.com, *"Service"*, http://developer.android.com/reference/android/app/Service.html, Retrieved 1. 2. 2013.

[12] Android.com, *"PowerManager"*, http://developer.android.com/reference/android/os/PowerManager.html, Retrieved 1. 2. 2013.

[13] Android.com, *"AlarmManager"*, http://developer.android.com/reference/android/app/AlarmManager.htm, Retrieved 1. 2. 2013.