Southeast Europe Journal of Soft Computing ISSN 2233 – 1859
Available online: http://scjournal.ius.edu.ba

Vol 10, No. 2, September 2021, pp. 28-32

MQTT And TCP Socket JavaScript Interface For Hybrid Mobile Applications

[1]Yusuf Dağlıoğlu, [1*]Ulviye Hacizade

[1]Faculty of Engineering, Halic University, Istanbul, Turkey
*Corresponding Author: ulviyehacizade@halic.edu.tr

Article Info

Article history:

Article received on 29 December 2021

Received in revised form 31 December 2021

Keywords:

MQTT; TCP; Android; Cordova; JavaScript

ABSTRACT: Today's needs require portable platforms such as mobile and IoT devices to be used more widely. The mobile platform development industry has grown rapidly as a result of the requirements of today's needs. There are various operating systems available on market such as iOS and Android. Naturally, software companies need to run their applications on multiple operating systems. Therefore, the software companies need to develop a new software with the native programming language of each operating system. Some frameworks like Cordova make possible to develop hybrid applications which based on web technologies and they are able to run on multiple operating systems. In some cases, developers may need to access device resources. In these cases, the developers should develop custom plugin for the hybrid framework. In this study, a plugin for Cordova framework has been developed that allows you to interact between web technologies and MQTT and TCP protocols through the JavaScript API.

1. INTRODUCTION

Hybrid applications are built on both native language of operating system and web technologies (HTML, CSS, JavaScript) [1]. Writing native applications for each operating system on the market has several disadvantages. Therefore, many developers choose to have web-view components for both mobile and desktop applications. But web technologies have limitations because of its nature and security reasons. SOP (Same Origin Policy) is an important example for security limitations [2]. Also, web technologies consume more resources than native applications because web technologies have additional layers (such as JavaScript engine and HTML renderer) to compile the web technologies on run-time.

There are many different frameworks like Apache Cordova [3], Phonegap [4] and ElectronJS (previously called Atom Shell) [5] which support web-view based applications. Those frameworks are using widely on different domains [5]. Those frameworks have many advantages for developers like bootstrapping new projects [6] and supporting plugins to access native API's of operating system. For example; WebView without any plugin only supports limited communication types like WebSockets and HTTP (AJAX) requests. But the developer may need to access one of the native API's of Message Queuing Telemetry Transport (MQTT) or Transmission Control Protocol (TCP) sockets. As a solution in this case, in this work, a plugin has been developed for access MQTT and TCP from mobile hybrid applications.

MQTT is lossless communication protocol which based on publish-subscribe model that transports messages between devices [7]. Subscriptions are related to topics. Each topic has its own message queue. The responsibility of queue management and message

delivery is on the server which called “broker” [7]. There are many broker implementations on the market. The broker sends the messages to client related to their subscriptions. Each client can be both subscriber or publisher.

TCP is the protocol on the “transport layer” [8] of the internet protocol suite [9]. It is low level protocol, for that reason its being use widely as communication protocol between devices.

2. RELATED WORKS

For developers, a push notification service is possible to use with many different protocols like WebSockets, XMPP. Each of them has its own advantages and disadvantages. But particular to MQTT is light-weight, therefore it is preferred on IoT devices. Carlos Silva [9] implemented a secure MQTT service to use for notification system to solve problem for instant pushing multiple messages. The server side of this implementation is served using Ubuntu and the connection between client and this server is secured by SSL. Both client and message broker which is Mosquitto, required extra configuration to secure communication. The client side implemented on iOS and Android as native applications. As MQTT library used Eclipse Paho and SwiftMQTT. The performance test was run both SSL and non-secure connection. On the results of those tests, the server used 6 times more CPU and RAM when SSL connection was used.

Tang et al. [10] developed a push notification system based on MQTT connection. The solution's client side implemented only on Android. The implementation tested running both server and clients 24 and 12 hour. The total network consumption of connection was too low.

Vitols et al. [11] studied the concerns of cross platform mobile applications architecture. One of the important considerations is that third party plugins for Cordova. Cordova framework is developing fast but most of plugins are not clear about the compatibility with Cordova framework. That poses risk and affects the efficiency of the developers because of compatibility problems. Kudo et al. [12] studied Cordova plugin attacks and presents a defense solution for them. Georgiev et al. [2] studied origin-based access control in hybrid mobile applications. The study demonstrates that frameworks does not properly compose the access-control policies. As various studies prove us that security policies should be define clearly especially on hybrid applications.

3. CORDOVA PLUGIN

In this work, Cordova plugin is written to communicate with TCP and MQTT connections directly from JavaScript inside any webview component. The Android part is implemented for this plugin. The architecture of the plugin is explained in Figure 1.

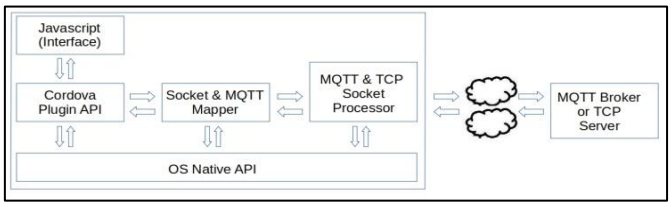

Figure 1: The architecture of plugin

JavaScript interface is the API which is provided for WebViews. This part works integrated with the Cordova plugin infrastructure and triggers the operating systems native methods. In the native part of extension, a mapping process is executed. This mapping process, detects the related TCP socket or MQTT broker instance. If it finds the related instance, it transfers the data with it. The connection instance should be passed on run-time from JavaScript interface. The connection instances are mapped by a connection-key which is a random generated number by Cordova plugin. Every connection-key has a mapped connection instance on operating system native part. If any unmapped key will be used, or any connection error will occur, native part of the Cordova plugin will throw an error. But in this case, JavaScript error callback function will be triggered and JSON data which includes error code will be passed to this callback function. Thus, the caller will be able to catch the exception from web-view. Therefore, API calls take both error and success callbacks.

The open source “paho” library has been used to manage the MQTT connections and Android operating systems embedded Socket library (“java.net.Socket” package) has been used to manage the TCP connections.

In this work, sample Android applications are written to use TCP and MQTT connections using the Cordova plugin. The first Android application has an graphical user interface to execute some calls to MQTT Cordova plugin. The user interface of this application is seen in Figure 2.

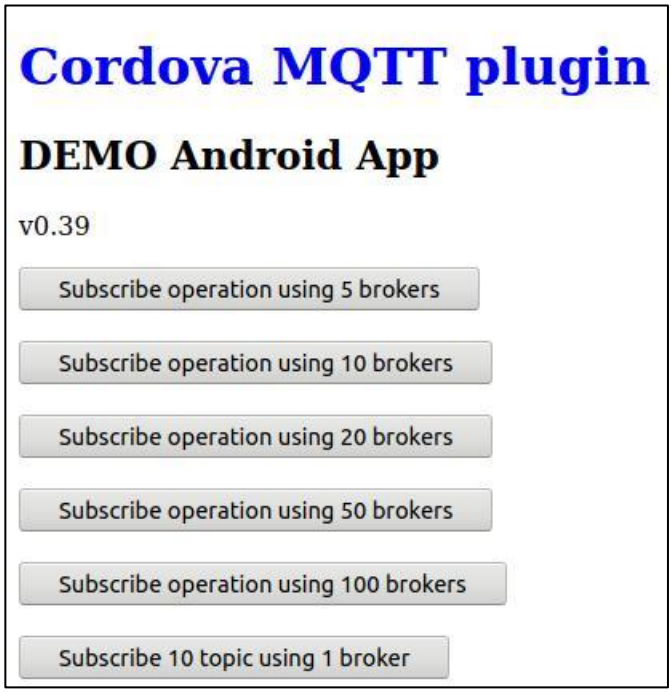

Figure 2: User interface of MQTT sample application

Many MQTT operations tested with this application and results have been examined with multiple Android emulators. The first emulator is "Nexus 4" which has 4GB RAM, x64 processor and it runs with Android 6.0 (API 23). The second emulator is a custom tablet which has 8GB RAM, x64 processor and runs Android 7.0 (API 24). The memory results according to operations are shown in Table 1 and Table 2.

Another application has been written for testing TCP connections via Cordova plugin. The user interface of this application is shown in Figure 3.

Figure 3: User interface of TCP sample application

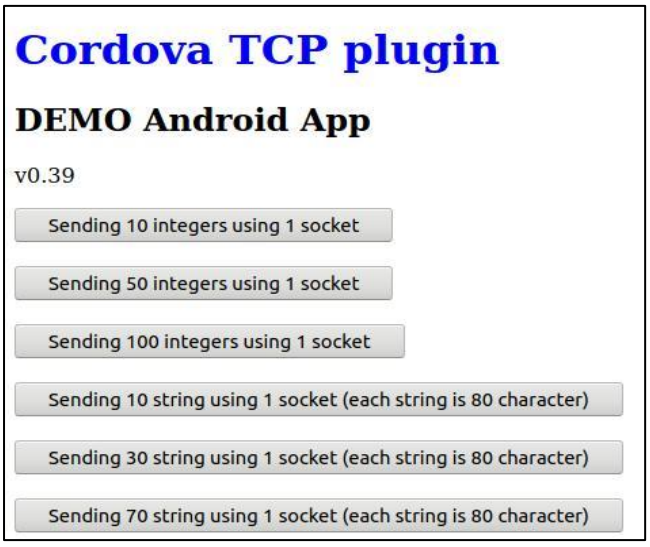

Table 1 Memory result of MQTT operations with Emulator-1

Operation	Total amount of memory (MB) before operation	Total amount of memory (MB) after operation
System idle	51	
Subscribe operation using 5 brokers	54	54
Subscribe operation using 10 brokers	57	57
Subscribe operation using 20 brokers	68	68
Subscribe operation using 50 brokers	79	79
Subscribe operation using 100 brokers	91	91
Subscribe 10 topic using 1 broker	55	55
Subscribe 20 topic using 1 broker	55	55
Subscribe 50 topic using 1 broker	55	55
Subscribe 100 topic using 1 broker	56	56
Sending 10 String transfer (each string is 80 character)	52	52
Sending 20 String transfer (each string is 80 character)	53	52
Sending 50 String transfer (each string is 80 character)	55	52
Sending 100 String transfer (each string is 80 character)	56	52
Sending 10 String transfer (each string is 300 character)	52	52
Sending 20 String transfer (each string is 300 character)	53	52
Sending 50 String transfer (each string is 300 character)	55	52
Sending 100 String transfer (each string is 300 character)	56	52

Table 2 Memory result of MQTT operations with Emulator-2

Operation	Total amount of memory (MB) before operation	Total amount of memory (MB) after operation
System idle	56	
Subscribe operation using 5 brokers	59	59
Subscribe operation using 10 brokers	62	62
Subscribe operation using 20 brokers	73	73
Subscribe operation using 50 brokers	84	84
Subscribe operation using 100 brokers	96	96
Subscribe 10 topic using 1 broker	60	60
Subscribe 20 topic using 1 broker	60	60
Subscribe 50 topic using 1 broker	60	60

Subscribe 100 topic using 1 broker	61	61
Sending 10 String transfer (each string is 80 character)	57	57
Sending 20 String transfer (each string is 80 character)	58	57
Sending 50 String transfer (each string is 80 character)	60	57
Sending 100 String transfer (each string is 80 character)	61	57
Sending 10 String transfer (each string is 300 character)	57	57
Sending 20 String transfer (each string is 300 character)	58	57
Sending 50 String transfer (each string is 300 character)	60	57
Sending 100 String transfer (each string is 300 character)	61	57

The memory results according to operations are shown in Table 3 and Table 4.

Table 3 Memory result of TCP operations with Emulator-1

Operation	Total amount of memory (MB) before operation	Total amount of memory (MB) after operation
System idle	53	
Sending 10 integers using 1 socket	54	54
Sending 50 integers using 1 socket	54	54
Sending 100 integers using 1 socket	54	54
Sending 10 string using 1 socket (each string is 80 character)	54	54
Sending 30 string using 1 socket (each string is 80 character)	54	54
Sending 70 string using 1 socket (each string is 80 character)	54	54
Sending 100 string using 1 socket (each string is 80 character)	54	54
Sending 10 string using 1 socket (each string is 300 character)	54	54
Sending 30 string using 1 socket (each string is 300 character)	54	54
Sending 70 string using 1 socket (each string is 300 character)	54	54
Sending 100 string using 1 socket (each string is 300 character)	54	54
Sending 50 String transfer (each string is 80 character)	60	57
Sending 100 String transfer (each string is 80 character)	61	57
Sending 10 String transfer (each string is 300 character)	57	57
Sending 20 String transfer (each string is 300 character)	58	57
Sending 50 String transfer (each string is 300 character)	60	57
Sending 100 String transfer (each string is 300 character)	61	57

Table 4 Memory result of TCP operations with Emulator-2

Operation	Total amount of memory (MB) before operation	Total amount of memory (MB) after operation
System idle	58	
Sending 10 integers using 1 socket	59	59
Sending 50 integers using 1 socket	59	59
Sending 100 integers using 1 socket	59	59
Sending 10 string using 1 socket (each string is 80 character)	59	59
Sending 30 string using 1 socket (each string is 80 character)	59	59
Sending 70 string using 1 socket (each string is 80 character)	59	59
Sending 100 string using 1 socket (each string is 80 character)	59	59
Sending 10 string using 1 socket (each string is 300 character)	59	59
Sending 30 string using 1 socket (each string is 300 character)	59	59
Sending 70 string using 1 socket (each string is 300 character)	59	59
Sending 100 string using 1 socket (each string is 300 character)	59	59
Sending 50 String transfer (each string is 80 character)	65	65
Sending 100 String transfer (each string is 80 character)	66	66
Sending 10 String transfer (each string is 300 character)	62	62
Sending 20 String transfer (each string is 300 character)	63	63
Sending 50 String transfer (each string is 300 character)	65	65
Sending 100 String transfer (each string is 300 character)	66	66

4 Conclusion

On mobile platforms, web-view without any plugin only supports limited communication types like WebSockets and HTTP (AJAX) requests. But the developer may need to access one of the native API's of Message Queuing Telemetry Transport (MQTT) or Transmission Control Protocol (TCP) sockets. As a solution in this case, in this work, a plugin has been developed for access MQTT and TCP from mobile hybrid applications. The MQTT and TCP sockets are accessed over the prepared JavaScript API which developed in this work. This JavaScript interface has been designed using Cordova framework's plugin infrastructure. In this design, data transferred over MQTT and TCP protocols to remote machines and the exchanged data passes to JavaScript side. Due to the fact that JavaScript supports limited data types, native object (such as "Socket", "Broker") of Java cannot be passed to JavaScript side. In this study; compatible

interfaces have been prepared so that native objects do not need to be passed. Also, these interfaces implemented in a way to catch exceptions. Memory usage has always been a problem in hybrid applications [N]. Because, on the one hand, the weight of web technologies negatively affects the application, on the other hand, plugin infrastructure that enables programmers to exchange data between JavaScript and on the native language also effects negatively by consuming more resources. In this study, during data exchange over TCP and MQTT protocols, the memory amount of the Android application has been examined. Considering the memory usage, the method of communicating over webview with both TCP and MQTT protocols does not consume more resources than expected, when evaluated in line with the performance expectations of an application using webview. When the amount of memory usage is examined, it is seen that the MQTT protocol consumes more memory than TCP communication. This situation is expected due to the nature of the MQTT protocol. As a result of the evaluations, there is no obstacles for data exchange using MQTT and TCP protocols via hybrid applications over plugin.

RFERENCES

[1] A. Khandeparkar, R. Gupta, B. Sindhya, “An Introduction to Hybrid Platform Mobile Application Development,” *International Journal of Computer Applications,* vol. 118, no. 15, p.31, 2015.

[2] M. Georgiev, S. Jana, V. Shmatikov, “Breaking and Fixing Origin-Based Access Control in Hybrid Web/Mobile Application Frameworks,” *Proc. NDDS Symposium*, The University of Texas at Austin, pp. 1-15, 2014.

[3] T. Hayit, A. Ozkan, “Determination of Student Readiness Level: A Mobile Application Study,” *International Journal of Education Science and Technology,* vol.3, no. 3, pp. 160-165, 2017.

[4] S. Kinney, S, *“Electron in Action,”* 1st edition, Manning Publications, 2018.

[5] C. Griffith, *"Mobile App Development with Ionic,"* 1st edition, O'Reilly Media, 2017.

[6] F. Yalçınkaya, H. Aydilek, M. Erten, N. İnanç, "IoT Based Smart Home Testbed Using MQTT Communication Protocol," *International Journal of Engineering Research and Development*, vol. 12, no. 1, pp. 317-324, 2020.

[7] D.E. Comer, “*Internetworking With TCP/IP Vol I: Principles, Protocols, and Architecture,”* Fourth Edition, Prentice Hall PTR, 2000.

[8] R. Braden, “Requirements for Internet Hosts - Communication Layers,” *Internet Engineering Task Force,* RFC: 1122, p. 81, 1989.

[9] C. Silva, R. Toasa, H.D. Martinez, J. Veloz, C. Gallardo, *"Secure Push Notification Service Based on MQTT Protocol for Mobile Platforms,"* School of Technology and Management, Polytechnic Institute of Leiria, 2017.

[10] K. Tang, Y. Wang, H. Liu, Y. Sheng, X. Wang, Z. Wei, "Design and Implementation of Push Notification System Based on the MQTT Protocol," *Proc. International Conference on Information Science and Computer Applications (ISCA)*, pp. 116-119, 2013.

[11] G. Vitols, I. Smits, A. Zacepins, "Issues of Hybrid Mobile Application Development with PhoneGap: A Case Study of Insurance Mobile Application," *Proc. 11th International Baltic Conference*, TUT Press ,pp. 215-220, 2014.

[12] N. Kudo, T. Yamauchi, T. H. Austin, "Access Control Mechanism to Mitigate Cordova PluginAttacks in Hybrid Applications," *Journal of Information Processing,* vol. 26, pp. 396-405, 2018.