DOI: https://doi.org/10.36719/2663-4619/112/284-289

Yusif Osmanov
Bakı Dövlət Universiteti
magistrant
https://orcid.org/0009-0001-9626-0004
osmanov.yusif@gmail.com

MQTT texnologiyalarının araşdırılması və tətbiqi

Xülasə

Message Queuing Telemetry Transport və ya MQTT, son dərəcə yüksək gecikmə və məhdud aşağı ötürmə qabiliyyəti olan əşyaların internet cihazları üçün nəzərdə tutulmuş rabitə protokoludur. Message Queuing Telemetry Transport maşından maşına (M2M) rabitə üçün mükəmməl protokoldur, çünki o, xüsusi olaraq aşağı bant genişliyi, yüksək gecikmə parametrləri üçün nəzərdə tutulmuşdur.

MQTT bir neçə cihaz arasında əlaqə yaratmaq üçün istifadə edilən sadə, yüngül mesajlaşma protokoludur. Bu, dərc-abunə modelinə əsaslanan TCP əsaslı protokoldur. Bu rabitə protokolu aşağı bant genişliyi və aşağı güc tələbləri olan resurs məhdud cihazlar arasında məlumat ötürmək üçün uyğundur. Beləliklə, bu mesajlaşma protokolu IoT Framework-də ünsiyyət üçün geniş istifadə olunur.

Açar sözlər: *Message Queuing Telemetry Transport (MQTT), əşyaların interneti, rabitə protokolu, bant genişliyi, yüksək gecikmə parametrləri, dərc-abunə modeli, TCP, resurs məhdud cihazlar, IoT Framework*

Yusif Osmanov
Baku State University
Master student
https://orcid.org/0009-0001-9626-0004
osmanov.yusif@gmail.com

Research and Application of MQTT Technologies

Abstract

Message Queuing Telemetry Transport (MQTT) is a communication protocol designed for Internet of Things (IoT) devices with high latency and limited bandwidth. MQTT is an ideal protocol for machine-to-machine (M2M) communication, as it is specifically designed to operate in low-bandwidth and high-latency environments.

MQTT is a simple and lightweight messaging protocol used to establish communication between multiple devices. It is a TCP-based protocol that follows the publish-subscribe model. This communication protocol is suitable for transmitting data between resource-constrained devices with low bandwidth and low power requirements. Therefore, MQTT is widely used for communication in IoT ecosystems.

Keywords: *Message Queuing Telemetry Transport (MQTT), Internet of Things, communication protocol, bandwidth, high latency settings, publish-subscribe model, TCP, resource-constrained devices, IoT Framework*

Giriş

MQTT-nin mövcud bant genişliyini maksimuma çatdırmaq məqsədi daşıyan dərc/abunə (pub/sub) ünsiyyət tərzi, son nöqtə ilə birbaşa əlaqə saxlayan adi müştəri-server arxitekturasına alter-nativdir. Bunun əksinə olaraq, mesajı ötürən müştəri (naşir) və onu qəbul edən müştəri və ya müştərilər (abunəçilər) pub/sub paradiqmasında bağlı deyillər. Üçüncü tərəflər - brokerlər - naşirlər və abunəçilər arasındakı əlaqələri idarə edirlər, çünki onlar bir-biri ilə birbaşa əlaqə saxlamırlar. Müştərinin mesajları dərc edib-etmədiyini və ya mesaj almaq üçün abunə olub-olmadığını göstərən nəşriyyatçılar və abunəçilər MQTT

müştərilərinə misaldır. Bu iki xüsusiyyəti yerinə yetirmək üçün eyni MQTT müştərisindən istifadə edilə bilər. Müştəri və ya cihaz məlumatı serverə və ya brokerə təqdim etmək istədikdə nəşr baş verir.

Tədqiqat

1. MQTT texnologiyalarının təsnifatı

Xidmətin Keyfiyyəti (QoS) Səviyyələri: Xidmətin keyfiyyəti müştəri ilə broker arasında razılaşma səviyyəsidir. Bu səviyyə müştərilərə mesajlarının çatdırılmasının etibarlılığını təmin etməyə kömək edir. 3 QoS səviyyəsi var (Veneri, & Capasso, 2024, s. 1059).

QoS 0 (Ən çoxu bir dəfə**):** Mesaj QoS səviyyəsi 0 ilə dərc edildikdə, mesaj ən çox bir dəfə çatdırılacaq. Başqa sözlə, mesajın çatdırılıb-çatdırılmamasına heç bir zəmanət yoxdur. Çatdırılırsa, maksimum bir dəfə çatdırılacaq. Bu QoS səviyyəsində göndərən mesajı yalnız bir dəfə dərc edir və sonra mesajı rədd edir. Bu ssenaridə etirafı gözləmək yoxdur. Beləliklə, əgər şəbəkə bağlanarsa, mesaj çatdırılmaya bilər. Şəbəkə sabitdirsə, mesaj yalnız bir dəfə çatdırılacaq.

Şəkil 1.1. QoS 0 diaqramı

Nə vaxt istifadə etməli?:

- Bir neçə mesajın itirilməsini ödəyə bilən proqramlar inkişaf etdirərkən.
- Yalnız bir dəfə dərc olunan mesajların uğurla çatdırılması üçün etibarlı əsas şəbəkəyə malik olan proqramları inkişaf etdirərkən.
-

Şəkil 1.2. Broker, naşir müştərisindən tək PUBLISH mesajı alır.

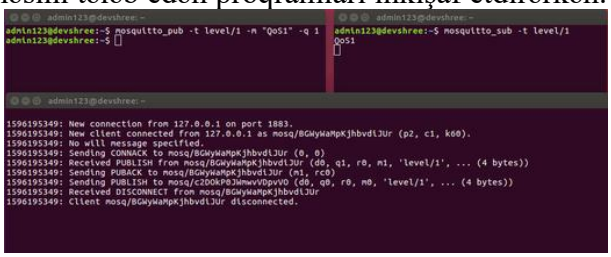
QoS 1 (Ən azı bir dəfə): Mesaj QoS səviyyəsi 1 ilə dərc edildikdə, mesaj ən azı bir dəfə çatdırılacaq. Bu QoS səviyyəsində göndərən mesajı dərc edir və həmçinin qəbuledicidən təsdiq alınana qədər onu saxlayır. Təsdiq alındıqdan sonra göndərən mesajı atacaq. Əgər bildiriş vaxtında alınmazsa, mesaj yenidən göndəriləcək. Beləliklə, bu QoS səviyyəsində mesajın uğurlu çatdırılmasını təmin etmək üçün bir dəfə və bəzən hətta bir dəfədən çox çatdırıla bilər.

Şəkil 1.3. QoS 1 diaqramı

Nə vaxt istifadə etməli?:

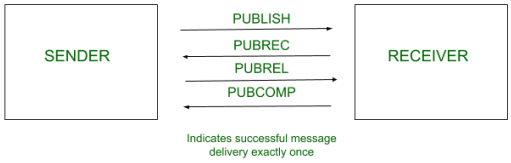
- Mesajların itkisini ödəyə bilməyən proqramlar inkişaf etdirərkən.
- Dublikat mesajların çatdırılmasını idarə edə biləcək proqramlar hazırlayarkən.
- Əlavə rabitə xərcləri istəməyən tətbiqlər inkişaf etdirərkən.

- Çox etibarlı əsas şəbəkəyə malik olmayan və buna görə də etibarlılıq tədbirlərinin protokol səviyyəsində həyata keçirilməsini tələb edən proqramları inkişaf etdirərkən.



Şəkil 1.4. Broker PUBLISH mesajını alır və PUBACK-ı naşir müştərisinə geri göndərir.

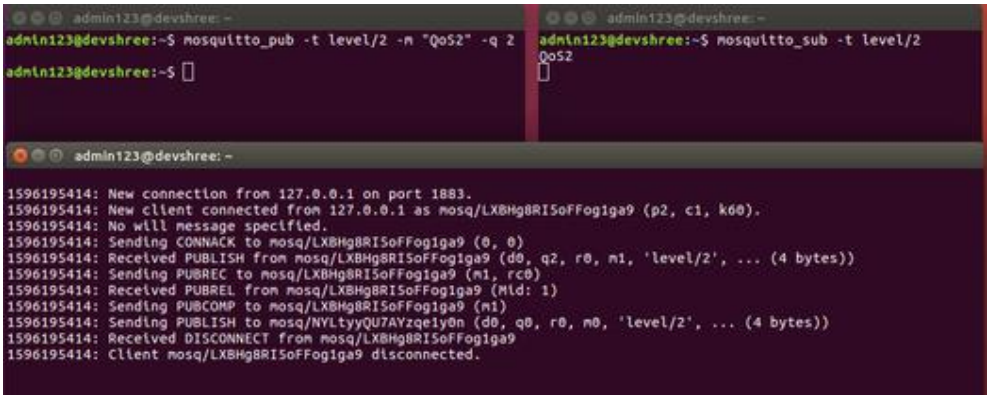
QoS 2 (Məhz bir dəfə): Mesaj QoS səviyyəsi 2 olan müştəri tərəfindən dərc edildikdə, mesaj tam olaraq bir dəfə çatdırılacaq. Bu QoS səviyyəsində göndərən mesajı dərc edir, saxlayır və PUBREC-i gözləyir. Alıcı mesajı aldıqdan sonra göndərənə PUBREC göndərir. PUBREC mesajın qəbulediciyə uğurla çatdırılmasının göstəricisidir. Beləliklə, PUBREC aldıqdan sonra mesaj göndərən tərəfindən atılacaq. Daha sonra PUBREL-i qəbulediciyə göndərir. PUBREL-i qəbul edərkən qəbuledici saxlanmış vəziyyətləri ləğv edir və PUBCOMP göndərir. Göndərən son PUBCOMP-u qəbul etdikdə, o, həmçinin əvvəllər saxlanmış vəziyyətləri ləğv edəcək. Beləliklə, QoS səviyyəsində 2-də ünsiyyət üçün istifadə olunan əlavə mesajlar mesajın tam bir dəfə uğurla çatdırılmasını təmin edir (Baruah, 2024).



Şəkil 1.5. QoS 2 diaqramı

Nə vaxt istifadə etməli?:

- Hər mesajın tam olaraq bir dəfə çatdırılmasını tələb edən proqramlar hazırlayarkən.
- Əlavə rabitə xərclərini ödəyə bilən proqramlar hazırlayarkən.
- Çox etibarlı əsas şəbəkəyə malik olmayan və buna görə də etibarlılıq tədbirlərinin protokol səviyyəsində həyata keçirilməsini tələb edən proqramları inkişaf etdirərkən.



Şəkil 1.6. Broker PUBLISH-i qəbul edir, PUBREC göndərir, PUBREL-i qəbul edir və sonra PUBCOMP-u naşir müştərisinə göndərir.

Nəşriyyatçı və Abunəçi QoS Səviyyələri arasındakı fərq: Naşir müştərisi (göndərən) və broker (qəbuledici) arasında mesaj rabitəsi üçün QoS səviyyəsi mesajın dərc olunduğu səviyyə ilə müəyyən ediləcək. Broker (göndərən) və abunəçi müştəri (qəbuledici) arasında mesaj rabitəsi üçün QoS səviyyəsi müştərinin mövzuya abunə olduğu səviyyə ilə müəyyən ediləcək.

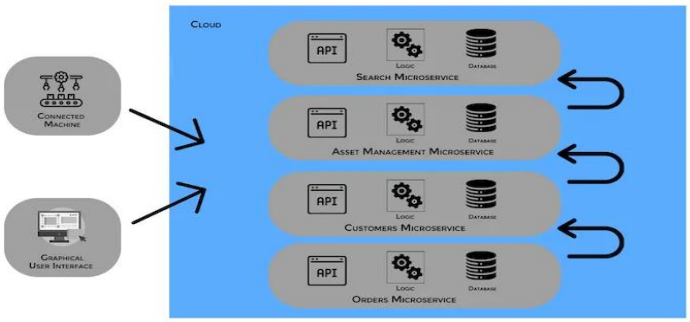
Rabitənin ümumi QoS səviyyəsi brokerin hər iki tərəfindəki iki QoS səviyyəsinin minimumuna bərabərdir. Bu ümumi səviyyə naşir müştərisi ilə abunəçi müştəri arasında mesajın çatdırılması zəmanətini müəyyən edir.

2. **MQTT texnologiyalarının IoT həllərinə tətbiqi**

IoT həlləri üçün MQTT protokolundan istifadə edən Hadisə Sürücülü Mikroservis Arxitekturası - Əşyaların İnterneti (IoT) bir çox təşkilat üçün prioritet ola bilər, lakin IoT layihələrinin böyük əksəriyyəti uğursuz olur. Cisco tərəfindən 2017-ci ildə yayımlanan bir hesabatda bu rəqəmin 74% olduğu göstərilib. IoT layihələrinin uğursuz olmasının bir çox səbəbi ola bilsə də, əsasən şirkətlərin işlək bir IoT həllini qurmaq üçün tələb olunan müxtəlif əlaqə və hesablama texnologiyalarının mürəkkəbliyini yetərincə qiymətləndirməməsindən qaynaqlanır.

Bu qədər hərəkət edən hissə ilə yaxşı düşünülmüş bir IoT həlli arxitekturasının olmaması layihənizin gözlənilən dəyəri çatdırmaqda uğursuz olmasına səbəb ola bilər. Bu uğursuzluq layihənin başlanğıcından icrasına və baxım mərhələsinə qədər istənilən mərhələdə baş verə bilər. IoT layihəsinin uğur qazanması üçün real-vaxt, çevik, genişləndirilə bilən və saxlanıla bilən bir sistemin inkişafına imkan verən arxitektur əsasın qurulması vacibdir. MQTT protokolundan istifadə edərək hadisə-sürücülü mikroservis arxitekturası yanaşması bu məqsədə çatmaq üçün düzgün seçimdir (Parikh, 2022, s. 272).

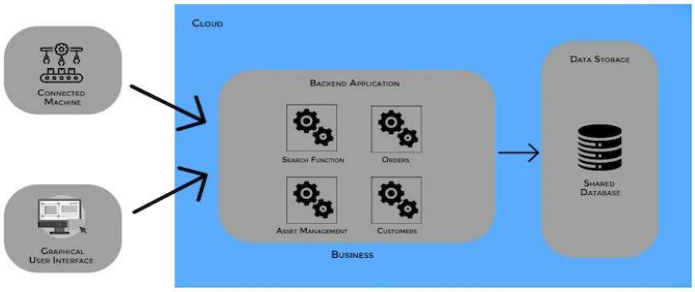
Mikroservis Arxitekturası Yanaşması - Bir IoT həllini və ya hər hansı bir proqram təminatını yaratmaq üçün daha ənənəvi bir yanaşma, istifadəçi interfeysi, biznes məntiqi və verilənlər bazasının bir serverdə yerləşən vahid bir proqramda birləşdirilməsidir (Hillar, 2018, s. 228).



Şəkil 2.1. Mikroservis Arxitekturası Yanaşması

Bu yanaşma bəzi hallarda işləyə bilsə də, tam funksional bir IoT həlli kimi mürəkkəb proqram təminatı sistemlərini yaratmaqda kifayət etmir. Əsasən, bu yanaşma miqyaslama və saxlanılma baxımından çətin olur və bir hissə işləmədikdə bütün xidmətlər dayanır.

Digər tərəfdən, Mikroservis Arxitekturası Yanaşması IoT tətbiqinizi kiçik funksional xidmətlər toplusu olaraq parçalamağa və hər bir xidməti öz proseslərində işlətməyə və API-lər vasitəsilə əlaqələndirməyə imkan verir.

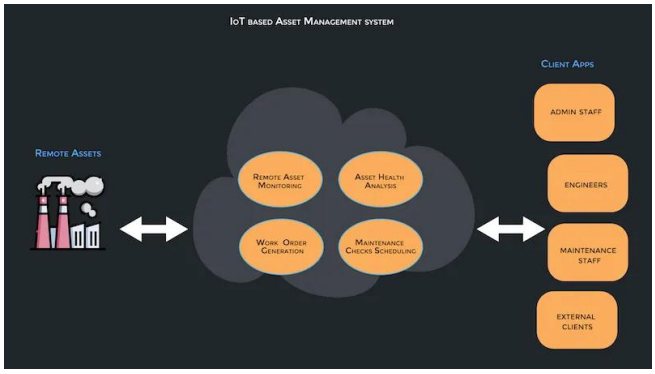


Şəkil 2.2. API-lər vasitəsilə əlaqələndirmə

Bu yanaşma elastikliyi artırır, məsələn, bütün mikroservis arxitekturasını yenidən yerləşdirmədən tək bir funksiyanın kodunu yeniləmək imkanı verir. Bundan əlavə, bu mikroservis kolleksiyaları böyük makro xidmətlərə birləşərək, tək bir funksiyanın kodunu ümumi bir xidmət və ya daha böyük son istifadəçi tətbiqində tez bir zamanda yeniləmək qabiliyyətini artırır.

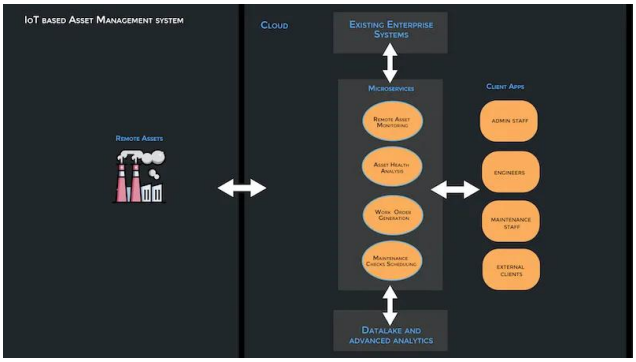
Mikroservislərin müstəqil yerləşdirilə bilməsi həm də IoT tətbiqinizin funksionallığını genişləndirməyə imkan verir – sistemi heç bir dayanma vaxtı olmadan, geniş miqyasda daha çox xidmət əlavə edərək təkmilləşdirmək mümkündür (Smart, 2020).

Məsələn, Uzaqdan Aktivlərin Monitorinqi, Aktivlərin Sağlamlıq Analizi, İş Sifarişlərinin Yaradılması və Baxım Yoxlamalarının Planlaşdırılmasından məsul olan IoT əsaslı Aktiv İdarəetmə sisteminin dizaynını nəzərdən keçirək. Sistem bulud əsaslıdır və funksiyalarını yerinə yetirmək üçün IoT cihazlarını özündə birləşdirir. Bu sistemin administrasiya heyəti, mühəndislər, texniki xidmət heyəti və xarici müştərilər üçün bir çox müştəri tətbiqi var (Dalmaris, 2023, s. 528).



Şəkil 2.3. IoT əsaslı Aktiv İdarəetmə sistemi

Əlavə olaraq, mikroservis arxitekturası Docker konteynerləri ilə birlikdə istifadə olunur və bu, IoT həllinizin hər bir funksiyası (mikroservis) üçün kodun müstəqil bir proqram paketi kimi yerləşdirilməsinə imkan verir. Bu proqram paketi onu işlətmək üçün lazım olan hər şeyi özündə cəmləşdirir və bu paket istənilən yerdə (buludda və ya daxili infrastruktura) yerləşdirilə bilər. Bu, IoT sisteminizin komponentlərinin nasazlıq nöqtələrinin bir-birindən daha müstəqil olması deməkdir. Əsasən, nasazlıq səthini təcrid etməklə daha sabit ümumi tətbiq arxitekturası yaradır. Bu, proqram təminatının inkişafını əhəmiyyətli dərəcədə sürətləndirir və IoT tətbiqinizi saxlamağı, yeniləməyi və genişləndirməyi asanlaşdırır. Yəni, bu nümunə üçün, biznes funksiyalarınıza xidmət etmək üçün dörd yeni tətbiq yazır və onları mövcud IT sistemlərinizlə API-lər vasitəsilə əlaqələndirirsiniz (Syed, 2024).



Şəkil 2.4. Docker ilə IoT əsaslı Aktiv İdarəetmə sistemi

İndi əsas məqam budur. Əşyaların İnterneti çox səviyyəli bir ekosistemdir və sensorlar və daxili proqram təminatı ilə təchiz olunmuş cihazlar telemetriya məlumatlarını toplayır, məlumatları daxili

və ya bulud əsaslı serverə ötürür və buna görə də IoT cihazları, mikroservislər, mövcud IT sistemləri və verilənlər bazaları, analitik həllər, vizuallaşdırma paneli və mobil tətbiqləri əhatə edən əlavə IoT qatları arasında çoxlu mesaj mübadiləsi baş verir.

Nəticə

Tədqiqatdan alınmış elmi nəticələr aşağıdakılardan ibarətdir:

- AMD, İntel və ARM fərqli platformalarda kiçik həcmli məlumatların ötrülməsini həm sürətli, həm də ardıcıl paketlərlə ötrülməsini təmin edir.
- MQQT texnologiyalarından istifadə edərək, IoT həllərində dəqiq və sürətli nəticələr əldə etmək olur.
- COAP texnologiyalarında olan RESTful əsaslı bütün proqram təminatının əsaslı şəkildə MQTT ilə daha müasir formatda yazılmasını təmin edir.

Ədəbiyyat

1. Baruah, B. (2024). *Evolve from infrastructure to innovation with SAP on AWS: Strategize beyond infrastructure for extending your SAP applications, data management*, IoT I, 465.
2. Dow, C. (2024). *Internet of things programming projects: Build exciting IoT projects using Raspberry Pi 5*, Raspberry Pi Pico, and Python, 458.
3. Veneri, G., & Capasso, A. (2024). *Hands-on industrial internet of things: Build robust industrial IoT infrastructure by using the cloud*, 1059.
4. Parikh, D. (2022). *Raspberry Pi and MQTT essentials*, 272.
5. Hillar, G. C. (2018). *Hands-on MQTT programming with Python*, 228.
6. Dalmaris, P. (2023). *Node-RED and Raspberry Pi Pico W: From basics to flows for sensors, automation, motors, MQTT, and cloud services*, 528.
7. Syed, A. (2024). *Supply chain software security: AI, IoT, and application security*, 533.
8. Smart, G. (2020). *Practical Python programming for IoT: Build advanced IoT projects using a Raspberry Pi 4, MQTT, RESTful APIs, WebSockets, and Python* 3, 516.