**Retele de calculatoare – proiect**

**Aplicatie pentru monitorizarea resurselor SO**

**(Client MQTT)**

**Studenti:**

Florea Calin, grupa 1310B

Magdalin Stefan, grupa 1310B

**Profesor:**

Nicolae-Alexandru Botezatu

**Introducere:**

MQTT(MQ Telemetry Transport) este un protocol de rețea de tip publicare-abonare care permite transportul de mesaje între dispozitive. Este ideal pentru conectarea la distanță a dispozitivelor care au capabilitati limitate impunand astfel constrangeri de resurse sau lățimea de bandă a rețelei este limitată.

**Istoric:**

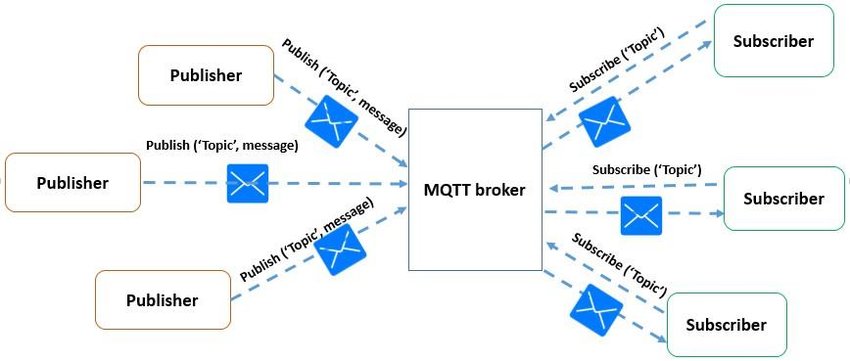
Andy Stanford-Clark și Arlen Nipper au fost autorii primei versiuni a protocolului MQTT în anul 1999. Scopul acestuia a fost de a avea un protocol care să fie eficient din punct de vedere al lățimii de bandă si care să folosească puțină energie deoarece dispozitivele erau conectate prin intermediul unei legături prin satelit, care, la acea vreme, era extrem de costisitoare.

Din punct de vedere istoric, „MQ” din „MQTT” provine din linia de produse MQ IBM MQ, unde înseamnă „Message Queue”. Cu toate acestea, protocolul oferă mesaje de publicare și abonare (fără cozi, în ciuda numelui). În specificația deschisă de IBM ca versiunea 3.1, protocolul a fost denumit „MQ Telemetry Transport”. După ce a preluat întreținerea standardului de la IBM, OASIS a lansat versiunea 3.1.1 pe 29 octombrie 2014. Pe data de 7 martie 2019 a fost lansata versiunea 5 a protocolului. Aceasta actualizare a fost una substantiala adaugand multe functii noi.



**Componente:**

1. Paradigma Publisher Subscriber : este modelul de baza al protocolului. Acesta ofera o alternativă la arhitectura tradițională client-server. Fata de modelul traditional in care doua dispozitive comunicau direct, acest concept introduce o a treia componenta, numita broker, care intermediaza trimiterea mesajului. Astfel, un dispozitiv publica un mesaj catre un topic, iar alt dispozitiv care este abonat la topicul respectiv va primi acel mesaj.



1. Mesajele : reprezinta informatia transmisa intre dispozitive
2. Broker : sarcina acestuia este să filtreze toate mesajele primite și să le distribuie corect abonaților.
3. Topic : se referă la un șir UTF-8 pe care brokerul îl folosește pentru a filtra mesajele pentru fiecare client conectat. Topicurile au o structura arborescenta, fiecare nivel fiind seaparat printr-o bara oblica(separator de nivel de subiect).



**Avantaje:**

Protocolului MQTT ajută la asigurarea unui transfer fluid al datelor cu lățime de bandă redusă și la reducerea sarcinii CPU și RAM. Printre avantajele MQTT față de protocoalele concurente se numără următoarele:

1. Transmisie eficienta a datelor
2. Fiind un protocol usor, faciliteaza implementarea rapida
3. Utilizare redusă a rețelei, datorită pachetelor de date minimizate
4. Distribuirea eficientă a datelor
5. Implementarea cu succes a controlului
6. livrarea rapidă și eficientă a mesajelor
7. utilizează cantități mici de putere, ceea ce este un avantaj pentru dispozitivele conectate
8. optimizează lățimea de bandă a rețelei

**Dezavantaje:**

Protocoalul MQTT poate presupune urmatoarele dezavanataje:

1. MQTT are cicluri de transmisie mai lente în comparație cu Protocolul de aplicare constrâns (CoAP)
2. MQTT este necriptat. În schimb, folosește TLS/SSL (Transport Layer Security/Secure Sockets Layer) pentru criptarea securității
3. Este dificil să creezi o rețea MQTT scalabilă la nivel global
4. Structura de subiecte a MQTT poate forma cu ușurință un arbore imens și nu există o modalitate clară de a împărți un arbore în domenii logice mai mici
5. nu există nicio modalitate prin care receptorul mesajului să știe cine a trimis mesajul inițial decât dacă aceste informații sunt conținute în mesajul real

**Mosquitto :**

Eclipse Mosquitto este un broker de mesaje care implementează versiunile de protocol MQTT 5.0, 3.1.1 și 3.1. Mosquitto este potrivit pentru utilizare pe toate dispozitivele, de la computere cu o singură placă de putere redusă până la servere complete. Protocolul MQTT oferă o metodă ușoară de a efectua mesaje folosind un model de publicare/abonare.

**Client MQTT**

**Mecanism KeepAlive :**

MQTT include o funcție de Keep Alive care oferă o soluție pentru problema conexiunilor pe jumătate deschise (sau cel puțin face posibilă evaluarea dacă conexiunea este încă deschisă). Keep alive asigură că legătura dintre broker și client este încă deschisă și că brokerul și clientul sunt conștienți că sunt conectați. Când clientul stabilește o conexiune cu brokerul, clientul îi comunică brokerului un interval de timp în secunde. Acest interval definește durata maximă de timp în care brokerul și clientul nu pot comunica între ei. Mecanismul Keep Alive este intervalul de timp maxim care este permis să treacă între punctul în care clientul termină transmiterea unui pachet de control și punctul în care acesta începe să trimită următorul pachet. Este responsabilitatea Clientului să se asigure că intervalul între pachetele de control trimise nu depășește valoarea Keep Alive. În absența trimiterii oricăror alte pachete de control, clientul TREBUIE să trimită un pachet PINGREQ.

Atâta timp cât mesajele sunt schimbate frecvent și intervalul de menținere în viață nu este depășit, nu este nevoie să trimiteți un mesaj suplimentar pentru a stabili dacă conexiunea este încă deschisă.

Dacă clientul nu trimite un mesaj în perioada de Keep Alive, acesta trebuie să trimită un pachet PINGREQ brokerului pentru a confirma că acesta este disponibil și pentru a se asigura că brokerul este încă disponibil.

Brokerul trebuie să deconecteze un client care nu trimite un mesaj sau un pachet PINGREQ într-un interval de timp egal cu 1,5 \* intervalul de Keep Alive. De asemenea, este de așteptat ca clientul să închidă conexiunea dacă nu primește un răspuns de la broker într-un interval de timp rezonabil.

Când brokerul primește un pachet PINGREQ, brokerul trebuie să răspundă cu un pachet PINGRESP pentru a arăta clientului că acesta este încă disponibil. De asemenea, pachetul PINGRESP nu conține încărcătură utilă.

**Mecanism Last Will and Testament (LWT) :**

Deoarece MQTT este adesea folosit în scenarii care includ rețele nesigure, este rezonabil să presupunem că unii dintre clienții MQTT din aceste scenarii se vor deconecta ocazional incorect. O deconectare incorectă poate apărea din cauza pierderii conexiunii, a bateriilor descărcate sau a multor alte motive. Un client se poate deconecta corect (cu un mesaj MQTT DISCONNECT) sau incorect (fără un mesaj MQTT DISCONNECT). Funcția Last Will and Testament oferă clienților o modalitate de a răspunde la deconectările neplăcute într-un mod adecvat.

Fiecare client poate specifica un mesaj Last Will and Testament (LWT) in mesajul CONNECT care inițiază conexiunea dintre client și broker. Brokerul stochează mesajul până când detectează că clientul s-a deconectat incorect. Ca răspuns la deconectarea greșită, brokerul trimite mesajul Last Will tuturor clienților abonați la topicul mesajului Last Will. Dacă clientul se deconectează corect cu un mesaj DISCONNECT corect, brokerul renunță la mesajul LWT stocat.

**Quality of Service (QoS) :**

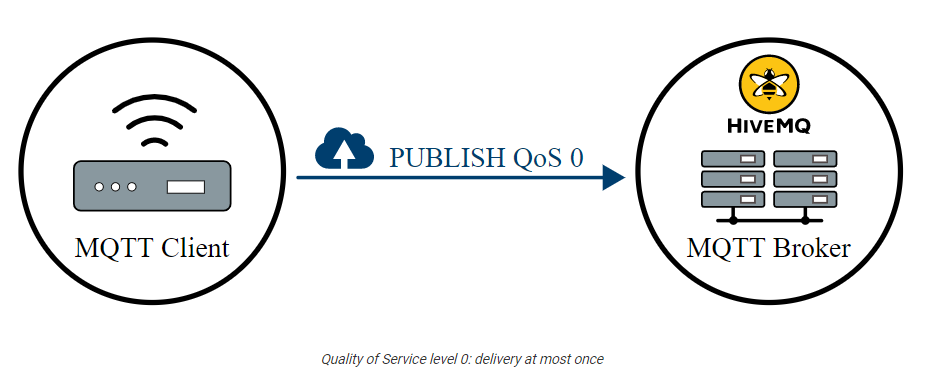
Quality of Service (QoS) este un acord între expeditorul unui mesaj și receptorul unui mesaj care definește garanția de livrare pentru un anumit mesaj. Există 3 niveluri QoS în MQTT:

* Cel mult o dată (QoS 0)
* Cel puțin o dată (QoS 1)
* Exact o dată (QoS 2).

Clientul care publică mesajul către broker definește nivelul QoS al mesajului atunci când trimite mesajul către broker. Dacă clientul abonat definește un QoS mai scăzut decât clientul de publicare, brokerul transmite mesajul cu QoS-ul cel mai scazut.

**Cel mult o dată (QoS 0) :**

Nivelul minim QoS este zero. Acest nivel de servicii garantează o livrare cu cel mai bun efort. Nu există nicio garanție de livrare. Destinatarul nu confirmă primirea mesajului, iar mesajul nu este stocat și retransmis de către expeditor. QoS 0 este adesea numit „fire and forget”.



**Cel putin o dată (QoS 1) :**

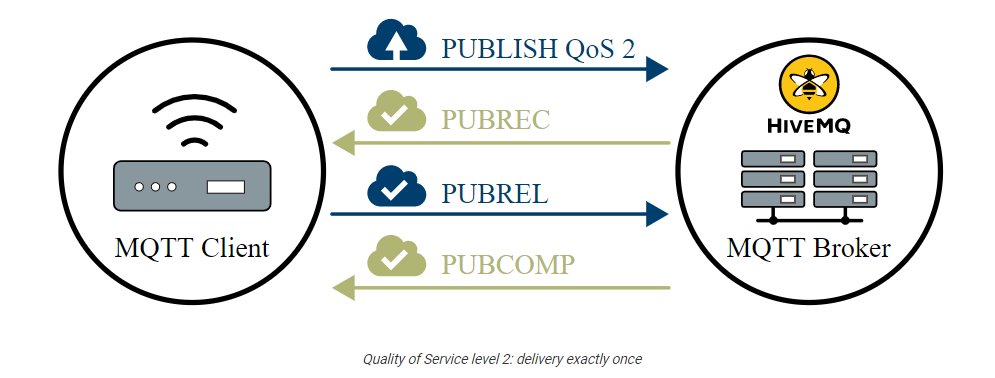
Nivelul QoS 1 garantează că un mesaj este livrat cel puțin o dată către receptor. Expeditorul stochează mesajul până când primește un pachet PUBACK de la receptor care confirmă primirea mesajului. Este posibil ca un mesaj să fie trimis sau livrat de mai multe ori. Dacă expeditorul nu primește un pachet PUBACK într-un timp rezonabil, expeditorul retrimite pachetul PUBLISH. Dacă clientul de publicare trimite mesajul din nou, acesta setează un flag duplicat (DUP). În QoS 1, acest flag DUP este utilizat doar în scopuri interne și nu este procesat de către broker sau client.



**Exact o dată (QoS 2) :**

QoS 2 este cel mai înalt nivel de serviciu din MQTT. Acest nivel garantează că fiecare mesaj este primit o singură dată de către destinatarii vizați. QoS 2 este cel mai sigur și cel mai lent nivel de calitate a serviciului. Garanția este oferită de cel puțin două fluxuri de cerere/răspuns între expeditor și destinatar. Când un receptor primește un pachet QoS 2 PUBLISH de la un expeditor, acesta procesează mesajul de publicare în consecință și îi răspunde expeditorului cu un pachet PUBREC care confirmă pachetul PUBLISH. Dacă expeditorul nu primește un pachet PUBREC de la receptor, acesta trimite din nou pachetul PUBLISH cu un steag duplicat (DUP) până când primește o confirmare. Odată ce expeditorul primește un pachet PUBREC de la receptor, expeditorul poate sterge în siguranță pachetul PUBLISH inițial. Expeditorul stochează pachetul PUBREC de la receptor și răspunde cu un pachet PUBREL. După ce receptorul primește pachetul PUBREL, poate renunța la toate stările stocate și poate răspunde cu un pachet PUBCOMP (același lucru este valabil atunci când expeditorul primește PUBCOMP). Până când receptorul finalizează procesarea și trimite pachetul PUBCOMP înapoi expeditorului, receptorul stochează o referință la identificatorul de pachet al pachetului PUBLISH original. Acest pas este important pentru a evita procesarea mesajului a doua oară. După ce expeditorul primește pachetul PUBCOMP, identificatorul de pachet al mesajului publicat devine disponibil pentru reutilizare.

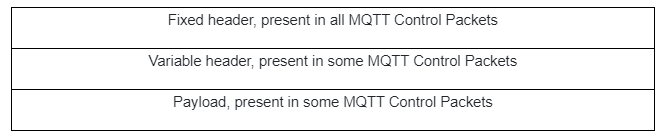
Când fluxul QoS 2 este complet, ambele părți sunt sigure că mesajul este livrat și expeditorul are confirmarea livrării.

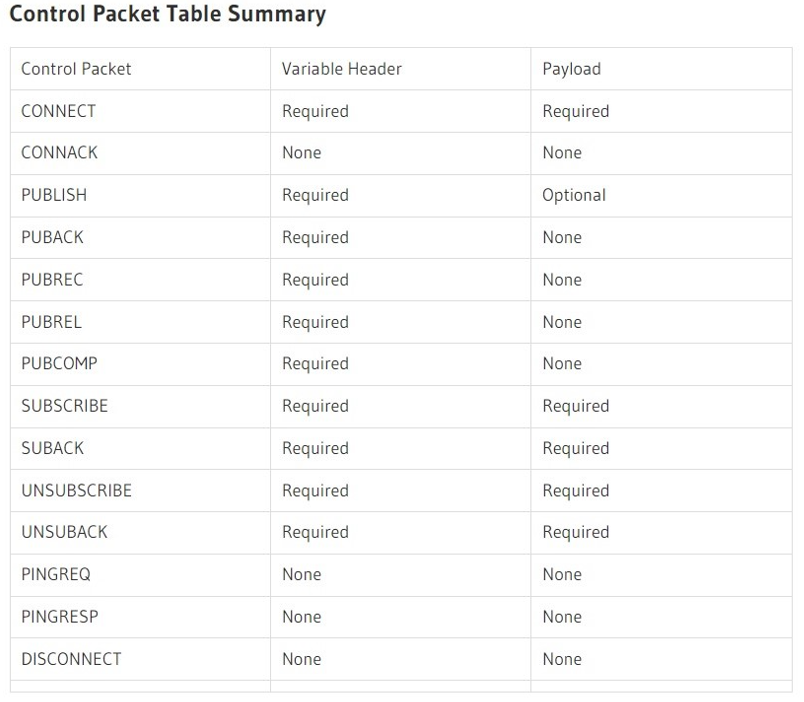


**Pachete de control :**

Protocolul MQTT funcționează prin schimbul unei serii de pachete de control MQTT într-un mod definit.

Un pachet de control MQTT este format din maxim trei parti întotdeauna în următoarea ordine:





**CONNECT**: pachet trimis de client catre server prin care acesta cere sa se conecteze

**CONNACK**: pachetul trimis de server ca raspuns la cererea clientului de conectare

**PUBLISH**: este trimis un mesaj pentru publicare de la client la server sau invers

**PUBACK**: raspunsul primit la un pachet PUBLISH cu QoS 1

**PUBREC**: prima parte a raspunsul primit la un pachet PUBLISH cu QoS 2 ( a doua parte a QoS 2)

**PUBREL**: a doua parte a raspunsul primit la un pachet PUBLISH cu QoS 2 ( a treia parte a QoS 2)

**PUBCOMP**: a treia parte a raspunsul primit la un pachet PUBLISH cu QoS 2 ( a patra parte a QoS 2)

**SUBSCRIBE**: este un pachet trimis de la client la sever si prin intermediul sau sunt create unul sau mai multe abonamente

**SUBACK**: pachetul trimis de la server la client prin care este confirmata primirea si procesarea pachetului SUBSCRIBE

**UNSUBSCRIBE**: este un pachet trimis de la client la sever prin intermediul sau clientul fiind dezabonat de la topicuri

**UNSUBACK**: pachetul trimis de la server la client prin care este confirmata primirea si procesarea pachetului UNSUBSCRIBE

**PINGREQ**: pachetul trimis de la un client la server prin care se verifica daca conexiunea este activa

**PINGRESP**: raspunsul primit de la un pachet PINGREQ, este trimis de la server la client prin care se semnaleaza ca serverul este activ

**DISCONNECT**: este pachetul trimis de la client la server prin care acesta notifica faptul ca acesta s-a deconectat corect de la broker

**Bibliografie:**

<https://docs.solace.com/MQTT-311-Prtl-Conformance-Spec/MQTT%20Control%20Packet%20format.htm#_Figure_2.1_->

<https://en.wikipedia.org/wiki/MQTT>

<https://www.hivemq.com/>

<https://mosquitto.org/>

<http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718063>