

Client MQTTv5

Davidoaia Alexandru-Ionut

I. Protocolul MQTT

1. Prezentare generala

MQTT este un protocol de transport al mesajelor Client-Server folosind arhitectura publish/subscribe, proiectat sa fie simplu de implementat, simplu de folosit și cu o dimensiune de pachete minima. Inițial proiectat pentru a gestiona conexiunile către conducte de petrol via satelit, protocolul este astăzi folosit în principal pentru IoT (Internet of Things). Principiile pe baza căruia funcționează sunt:

- Implementare simpla
- Ușor și eficient din punctul de vedere a lungimii de date (bandwidth)
- Data agnostic (datele transmise pot fi de orice format sau din surse multiple)
- QoS (quality of service - calitate a serviciilor) în livrarea datelor
- Continuous session awareness (poate notifica atunci cand are loc o pierdere de conexiune anormala)

2. Concepte cheie

Client/Broker

Un Client MQTT este orice dispozitiv ce se conecteaza la un Broker MQTT pentru a transmite si receptiona anumite date.

Un Broker MQTT este responsabil pentru receptionatul, filtratul, distribuitorul la fiecare abonat si trimisul mesajelor la unul sau mai multe conexiuni concurente de la Client-uri MQTT. In plus, monitorizează sesiunile fiecărui Client persistent, inclusiv topicurile la care este abonat și mesajele ratate, dar mai este și responsabil de autentificarea Clientilor.

Publish/Subscribe

Orice Client MQTT poate fi Publisher si/sau Subscriber. Un Publisher este cel care publica anumite date pe topic-uri dedicate, iar un Subscriber este cel care primește datele publicate pe topic-urile la care este abonat.

Un topic este un String UTF-8 folosit de broker pentru a filtra mesajele destinate fiecărui client. El este format din șiruri de caractere delimitate de caracterul “/” care separa nivelele topic-urilor, de exemplu “myhome/ground floor/+/temperature”, cu caracterul “+” reprezentand un wildcard pentru un singur nivel. Un Client abonat la acest topic va primi informațiile legate de temperatura tuturor dispozitivelor din parterul unei case. Evident, dispozitivele care transmit aceste informații sunt și ele Client-uri MQTT.

QoS

Quality of Service reprezintă nivelul garanției de livrare al unui mesaj, agreeat între cel care trimite mesajul și cel care îl recepționează.

Publisher-ul trimite un mesaj către Broker(server) cu un anumit nivel de QoS. Broker-ul, trimite, mai apoi, mesajul către Subscriber cu nivelul de QoS ales de acesta în timpul procesului de Subscribing. Dacă Subscriber-ul a ales un nivel mai scăzut de QoS decât cel transmis de Publisher, Broker-ul va trimite mesajul cu nivelul de QoS mai mic.

Nivelul QoS-ului(0,1 sau 2) trebuie ales corect în funcție de cerințele aplicației și de fiabilitatea rețelei.

QoS 0 -> „at most once”

Nivelul cel mai scăzut, QoS 0, oferă o încercare de livrare a mesajului cât mai bună, însă unde expeditorul nu se așteaptă ca aceasta încercare să fie mereu cu succes. Aceasta înseamnă că destinatarul nu confirmă primirea mesajului, iar expeditorul nu îl stochează sau retransmite. QoS 0, cunoscut în mod comun ca „fire and forget”, funcționează asemănător cu protocolul TCP basic, unde mesajul este trimis fără urmărire sau confirmare ulterioară.

Un exemplu unde s-ar putea folosi QoS 0 ar fi un sistem de monitorizare a temperaturii care primește un input la fiecare câteva secunde și care nu are nevoie de 100% din datele primite pentru a funcționa corect.

QoS 1->„at least once”

În QoS 1, accentul este pe asigurarea livrării mesajului cel puțin o dată către receptor. Când un mesaj este publicat cu QoS 1, expeditorul păstrează o copie a mesajului până când primește înapoi un packet de tip PUBACK de la receptor, confirmând astfel primirea cu succes.

Dar, dacă expeditorul nu primește acest packet PUBACK în timp util, el păstrează copia mesajului care dorește să fie trimis ca să asigure transmiterea cu succes a acestuia.

În QoS 1, dacă Publisher-ul trimite din nou același mesaj, se setează un duplicate flag(DUP), dar acesta nu este gestionat nici de client și nici de broker, fiind utilizat doar în scopuri interne.

Indiferent de DUP flag, receptorul trimite un packet PUBACK pentru a confirma primirea mesajului, astfel expeditorul este conștient de livrarea acestuia cu succes.

QoS 1 este considerat default.

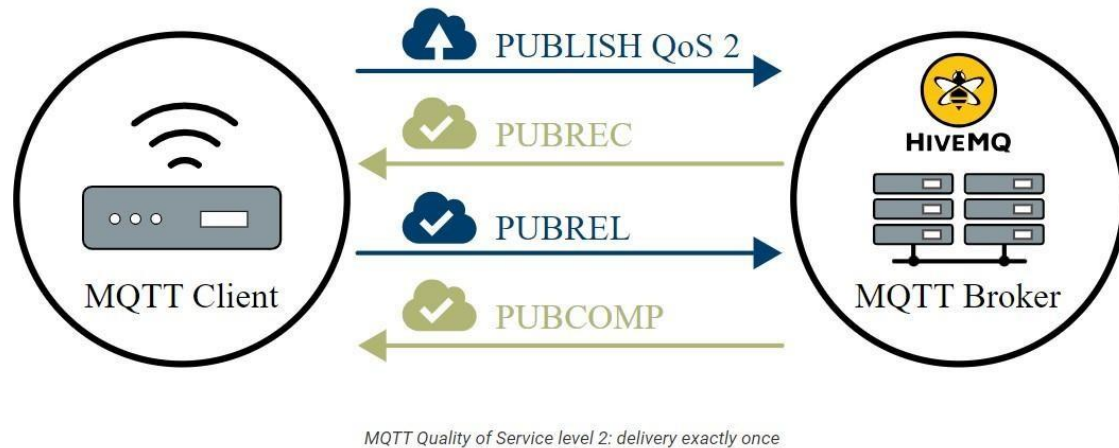
QoS 2->”Exactly once”

QoS 2 oferă cel mai înalt nivel de calitate al serviciilor în MQTT, asigurând ca fiecare mesaj trimis va livra o singură dată către destinatarul vizat. Pentru a realiza acest lucru, QoS 2 implică un sistem de 4 etape între expeditor și destinatar. În prima fază, în momentul în care un receptor primește un packet QoS 2 PUBLISH de la expeditor, acesta răspunde cu un packet PUBREC care confirmă procesarea cu succes a acestuia. Însa, dacă expeditorul nu primește acest pachet de la receptor, expeditorul re-trimite un packet PUBLISH care are și un DUP flag.

În următoarea fază, după ce expeditorul primește PUBREC packet-ul, îl desface și îl stochează, iar mai apoi răspunde cu un alt packet de tip PUBREL.

După ce receptorul primește PUBREL packet-ul, acesta răspunde cu un PUBCOMP packet. Înainte ca receptorul să finalizeze procesarea și trimiterea PUBCOMP packet-ului înapoi expeditorului, acesta, receptorul, stochează o referință la identificatorul de pachet al PUBLISH packet-ului original. Acest pas este important pentru a evita procesarea mesajului a doua oară.

Dacă un pachet se pierde pe parcurs, expeditorul este responsabil să retransmită mesajul în timp util. Acest lucru se aplică și dacă expeditorul este un client MQTT sau un broker MQTT. Receptorul are responsabilitatea de a răspunde la fiecare mesaj de comandă în mod corespunzător.



KeepAlive

Keep Alive este o caracteristică a protocolului MQTT care permite unui client MQTT să-și mențină conexiunea cu un broker, trimițând pachete regulate de control numite PINGREQ către broker.

Când un client stabilește o conexiune cu un broker MQTT, acesta stabilește o valoare Keep Alive, care este un interval de timp exprimat în secunde. Clientul trebuie să trimită un pachet PINGREQ brokerului cel puțin o dată în acest interval pentru a detecta prezența și a menține conexiunea în funcțiune. La primirea unui pachet PINGREQ, brokerul răspunde cu un pachet PINGRESP, confirmând că conexiunea este încă activă.

Bit	7	6	5	4	3	2	1	0
byte 9	Keep Alive MSB							
byte 10	Keep Alive LSB							

LastWill

MQTT lastWill este un set de parametri menit să informeze Subscriber-ii cu privire la un anumit subiect despre deconectarea anormală a unui anumit client. Clientul nu poate trimite un mesaj atunci când se deconectează în mod neașteptat, așa că îi cere brokerului să-l trimită în numele său.

Odată ce brokerul detectează o întrerupere a conexiunii, publică un mesaj predefinit, așa-numitul lastWillMessage, la subiectul predefinit, astfel încât Subscriber-ii să fie informați despre deconectare.

Lista parametrilor funcției Last Will in MQTT este următoarea:


lastWillMessage -> un mesaj care va fi trimis Subscriber-ilor unui anumit topic în cazul deconectării unui client.

lastWillTopic -> un subiect MQTT pentru a publica lastWillMessage.

lastWillRetain -> o valoare booleană care indică dacă mesajul LastWill trebuie reținut sau nu. Dacă este reținut, mesajul LastWill va fi trimis tuturor noilor abonați ai lastWillTopic. Dacă nu, noii abonați nu vor primi mesajul LastWill.

lastWillQoS -> un nivel de QoS care sa fie folosit la trimiterea lastWillMessage

Configurarea Last Will se realizeaza in momentul conectării la un broker MQTT cu un CONNECT packet.

CONNECT		MQTT-Packet
clientId		„client1”
cleanSession		true
username		„user1”
password		„pw1”
lastWillTopic		„/car/door”
lastWillQos		0
lastWillMessage		„connection disabled”
lastWillRetain		false
keepAlive		120

3. Structura pachete

Pachetele de control MQTT sunt formate din trei parti: un Header Fixat, un Header Variabil (doar în unele pachete), si un Payload (doar în unele pachete).

Header Fixat
Header Variabil
Payload

Header-ul Fixat are dimensiune de 1 byte, cu biții 7-4 fiind folosiți pentru a denota tipul pachetului (16 tipuri posibile), iar biții 3-0 pentru a reprezenta flag-urile specifice fiecărui tip de pachet de control MQTT. În restul de octeți vor fi conținute restul de date din pachet, inclusiv Header-ul Variabil si Payload-ul.

Bit	7	6	5	4	3	2	1	0
byte 1	Tip pachet MQTT				Flag-uri specifice fiecarui pachet MQTT			
byte 2...	Lungime ramasa							

Header-ul variabil este conținut doar în unele tipuri de pachete MQTT. Conținutul sau variază după tipul de pachet. Multe tipuri de pachete (PUBLISH cand QoS > 0, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK) contin campul “Packet Identifier”, cu lungimea de doi octeți, La fel, multe dintre pachete (CONNECT, CONNACK, PUBLISH, PUBACK, PUBREC, PUBREL, PUBCOMP, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK, DISCONNECT, AUTH) contin un camp de proprietati, de lungime variabila.

Ultima componenta a multor pachete MQTT este acea de Payload, un camp necesar in pachetele CONNECT, SUBSCRIBE, SUBACK, UNSUBSCRIBE, UNSUBACK si optional in PUBLISH. Conținutul și lungimea acestui camp variază de la pachet la pachet.

Câmpurile text din pachetele de control MQTT sunt codificate ca string-uri UTF-8, iar aceste string-uri sunt prefixate de un camp cu lungimea de doi octeți care conține numărul de octeți din string. Perechile name-value sunt reprezentate prin perechi de string-uri codificate UTF-8, cu primul string servind ca numele, iar al doilea ca valoarea.

II. Implementare

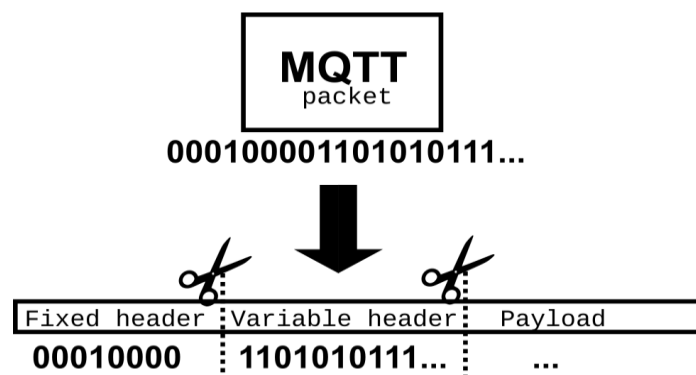
Proiectul va fi implementat în limbajul Python, folosind biblioteca **Tkinter** pentru a implementa interfața grafică și **sockets** pentru networking. Funcționalitatea minimă necesară a clientului va fi:

- Sa deschida conexiunea către un server
- Să publice mesaje în care ar putea fi interesați alți clienți
- Sa se aboneze pe anumite topic-uri pentru a cere anumite mesaje
- Sa anuleze abonamentul către un topic
- Sa închida conexiunea cu serverul

Interfața va oferi abilitatea de a configura adresa Brokerului, ID-ul clientului și mesajul LastWill pentru când conexiunea va fi oprită. Parametrii sistemului de calcul (temperatura CPU, încărcare CPU/RAM etc.) vor fi monitorizați și publicați periodic pe topic-uri dedicate (implementarea acestei funcționalități variază în funcție de sistemul de operare, deci aplicația va fi cel mai probabil proiectată să ruleze *ori* pe Linux, *ori* pe Windows.)

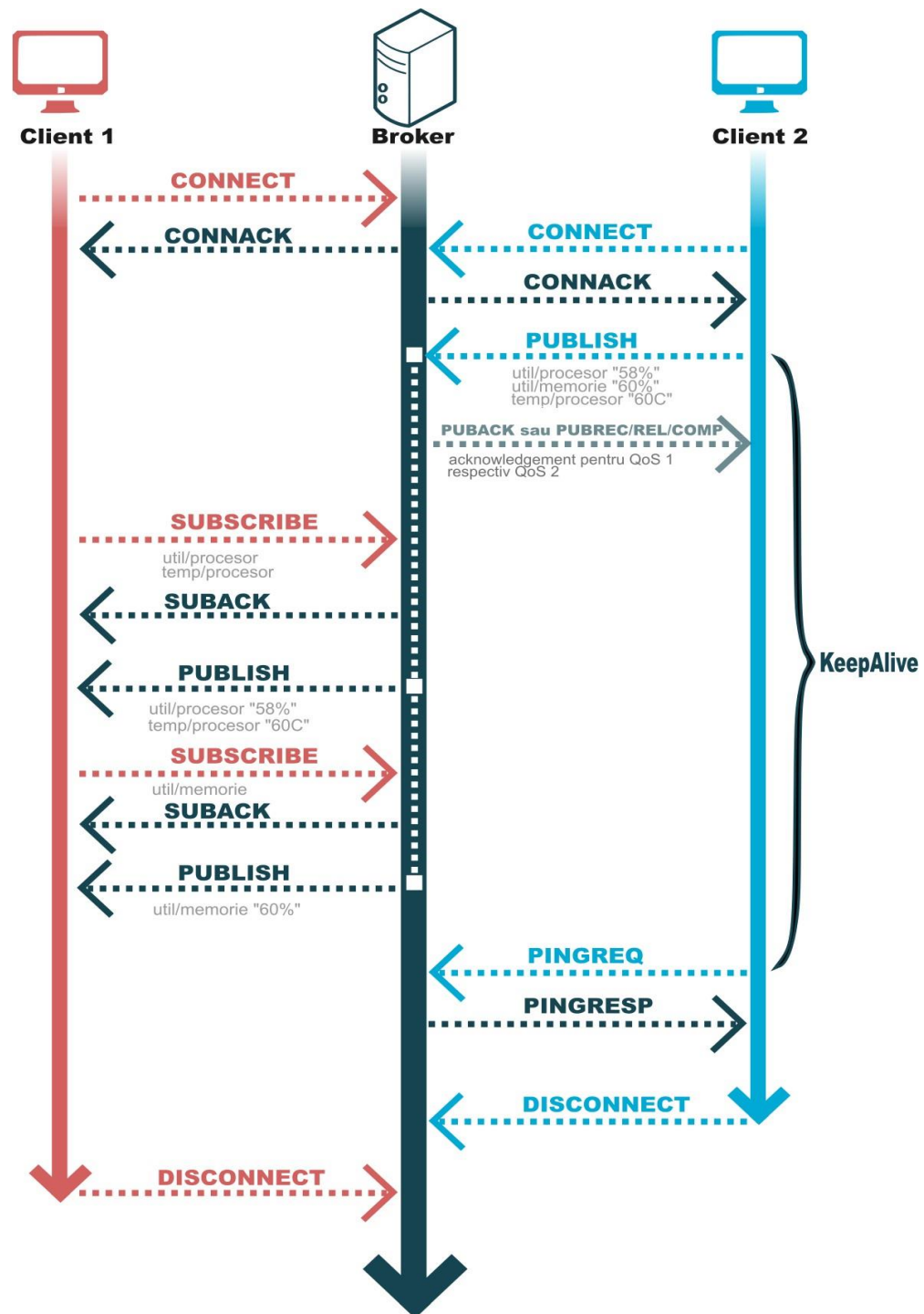
Programul va trebui să aibă abilitatea de a transmite și recepționa pachetele corespunzătoare, transportate prin intermediul protocolului TCP, folosindu-se de uneltele oferite de Tkinter pentru a stabili o conexiune cu Brokerul. Pentru a nu apărea probleme de delay sau de blocare, programul va utiliza threading, cu cel puțin două fire de execuție: unul fiind rezervat pentru interfața, și încă unul pentru partea de networking (conexiunea la un Broker).

Va fi necesară abilitatea de a decoda și codifica diferite tipuri de date (Byte, Two/Four/Variable Byte Integer, UTF-8 String / String Pair, etc.) pentru a extrage informațiile necesare și transmite informațiile în formatul corect.



III. Schema de conexiune

1. Functionarea generala a unei retele MQTT



2. Funcționarea aplicației Client (Pentru QoS 0)

