Universitatea Tehnică „Gheorghe Asachi“ din Iași
Facultatea de Automatică și Calculatoare

# BROKER MQTT v5
## Proiect - Rețele de Calculatoare

Profesor coordonator
S.I.dr. Nicolae Botezatu

Studenti
Sfichi Alin-Ionuț
Pitic Emanuel

# Cuprins

# Introducere

## 1. Ce este un protocol?

Un protocol reprezintă un set de reguli și instrucțiuni care permit calculatoarelor să comunice și să transmită pachete de date între ele. Ca o comparație simplă, este asemănător cu un gest de strângere a mâinii înainte de a începe o conversație – respectând anumite reguli de interacțiune.

## 2. Ce este MQTT?

MQTT este un protocol de transport al mesajelor, utilizând modelul Client-Server, bazat pe mecanismul publish/subscribe. Datorită simplității sale și a eficienței în utilizarea resurselor, precum și a capacității de a funcționa în condiții de semnal slab, acesta este ideal pentru diverse situații, inclusiv în medii restrânse, cum ar fi comunicarea între dispozitive (M2M) și proiecte IoT, care necesită un cod redus. În prezent, multe aplicații renumite adoptă acest protocol, printre care se enumeră Facebook Messenger și Instagram.

Pentru a înțelege mai bine acest protocol, vom analiza câteva concepte cheie:

- Publish / Subscribe
- Broker
- Topicuri
- Mecanisme precum:
    1. Keep Alive
    2. Last Will and Testament
    3. Quality of Service (QoS)

# Publish/Subscribe

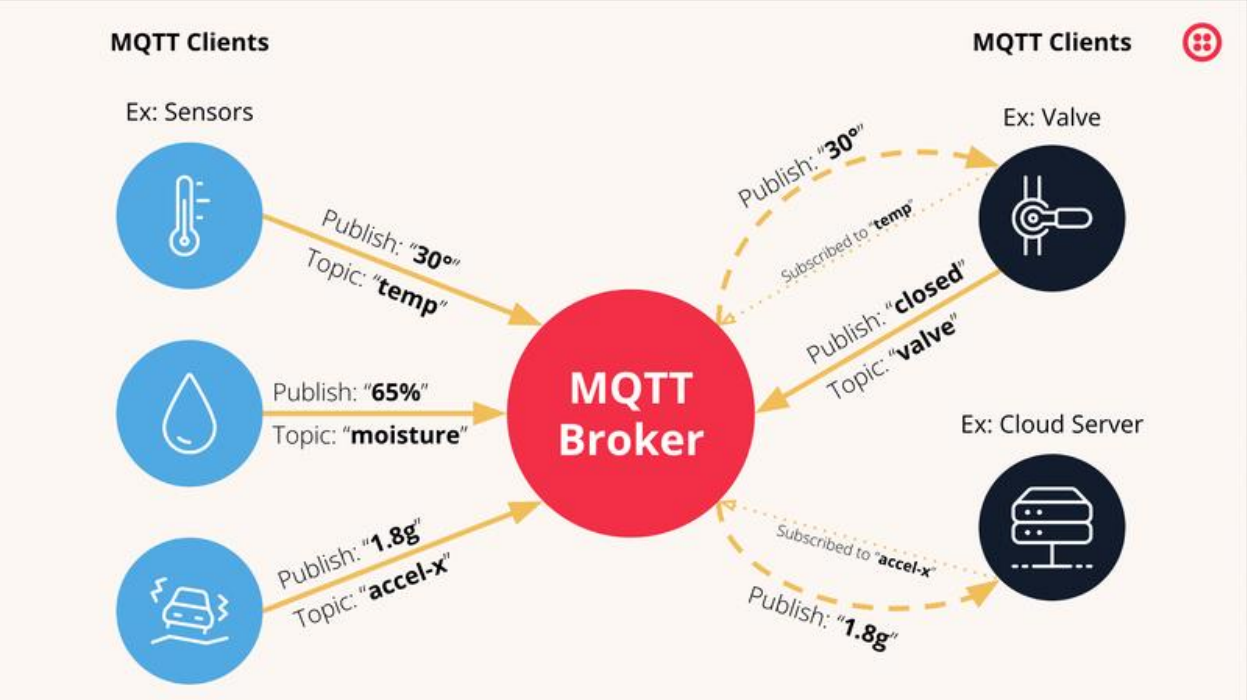
Modelul Publish/Subscribe are 2 componente de baza.

1. Publisher

Publisherii sunt entitățile care generează și transmit mesaje. Fiecare mesaj este asociat unui anumit "topic", pe care îl vom explica mai târziu. În contextul protocolului MQTT, clienții joacă rolul de publisher atunci când trimit mesaje către broker.

2. Subscribers

Subscribers sunt entitățile care își manifestă interesul de a primi anumite mesaje, abonându-se la unul sau mai multe topicuri. După abonare, primesc mesajele publicate pe aceste topicuri. În MQTT, clienții devin subscribers atunci când stabilesc o conexiune cu broker-ul, specificând topicul de interes.



# Broker

Un broker MQTT acționează ca o punte de comunicare între dispozitivele client și joacă un rol central în funcționarea protocolului. În versiunea MQTT v5, brokerul a primit îmbunătățiri pentru a oferi funcționalități suplimentare și a răspunde mai bine nevoilor actuale.

1. **Gestionarea conexiunilor**
   Brokerul primește conexiuni de la clienți, care inițial trimit un pachet CONNECT. În versiunea v5, pachetul CONNECT include proprietăți suplimentare, cum ar fi intervalul de expirare a sesiunii sau proprietăți specifice utilizatorului.
2. **Gestionarea abonamentelor**
   Brokerul păstrează o evidență a tuturor abonamentelor. Când un client dorește să se aboneze la un topic, trimite un pachet SUBSCRIBE, iar brokerul răspunde cu SUBACK. În MQTT v5, SUBACK poate include motive clare pentru succesul sau eșecul procesului de abonare.
3. **Publicarea și rutarea mesajelor**
   Când un client publică un mesaj, brokerul determină destinatarii pe baza abonamentelor existente. Pachetele PUBLISH din MQTT v5 pot conține un "alias" de topic, pentru a minimiza dimensiunea pachetului în cazul în care sunt utilizate frecvent topicuri lungi.
4. **Gestionarea calității serviciului (QoS)**
   Brokerul asigură livrarea mesajelor în funcție de nivelul QoS solicitat, care poate fi 0 (cel mult o dată), 1 (cel puțin o dată) sau 2 (exact o dată).
5. **Funcționalități avansate în MQTT v5**:
   - **Shared Subscriptions**: Permite ca mai mulți clienți să se aboneze la același topic, mesajele fiind distribuite echilibrat între membrii grupului.
   - **Controlul fluxului**: Clientul poate specifica câte mesaje poate primi înainte de a trimite un ACK.
   - **Proprietăți suplimentare**: Adaugă metadate la pachete, cum ar fi motivele pentru anumite acțiuni sau durata de expirare a mesajelor.
6. **Securitate și autentificare**
   Majoritatea brokerilor MQTT v5 oferă metode de autentificare prin nume de utilizator/parolă, certificat de client și metode mai avansate, precum autentificarea OAuth sau JWT.

# Topics

În protocolul MQTT, **topicurile** reprezintă canalele prin care mesajele sunt trimise și primite, jucând un rol esențial în organizarea și gestionarea comunicării între dispozitive.

1. **Structura**:
    a. Un topic este un șir de caractere care definește calea mesajului și este împărțit în mai multe nivele, separate prin /. De exemplu, casa/living/temperatura poate reprezenta un mesaj trimis de la un senzor de temperatură din camera de zi.

    b. Ierarhia topicurilor este definită de fiecare nivel. În exemplul anterior, casa este categoria principală, living un subnivel care specifică locația, iar temperatura definește tipul de date transmise.

2. **Abonarea și Publicarea**:
    a. Clienții se "abonează" la un topic pentru a primi mesaje. De fiecare dată când un alt client publică un mesaj pe acel topic, toți abonații vor primi automat mesajul.

    b. Este posibil să te abonezi la mai multe topicuri simultan, iar pentru topicuri complexe sau necunoscute se pot folosi wildcard-uri. De exemplu, casa/living/# va capta toate mesajele care includ casa/living, indiferent de ce urmează după.

3. **Wildcard-uri**:
    a. Wildcard-ul # (hash) permite captarea tuturor mesajelor dintr-o ierarhie. De exemplu, casa/# va include toate topicurile de sub casa, cum ar fi casa/living sau casa/bucatarie/temperatura.

    b. Wildcard-ul + (plus) permite abonarea la un singur nivel din ierarhie. De exemplu, casa/+/temperatura va capta toate mesajele de tip temperatură din orice cameră din casă.

4. **Siguranța și Accesul**:

    Pentru a menține securitatea și integritatea mesajelor, brokerul MQTT poate impune reguli stricte cu privire la cine poate publica sau se poate abona la anumite topicuri. Aceste controale sunt esențiale în aplicațiile industriale sau în rețele IoT sensibile, unde accesul la date trebuie restricționat.

5. **Topicuri cu Retenție**:

   Dacă un mesaj este publicat pe un topic cu opțiunea de retenție activată, brokerul va păstra acel mesaj ca fiind cel mai recent trimis pe acel topic. Astfel, un nou client care se abonează la acel topic va primi imediat mesajul reținut, oferindu-i acces la ultimele date disponibile, chiar dacă acestea au fost publicate anterior.

6. **Persistența mesajelor**:

   Mesajele reținute pot fi folosite pentru a trimite informații importante, cum ar fi starea unui dispozitiv sau a unui sistem. În situații în care un nou client se conectează pentru prima dată sau reconectează după o deconectare, brokerul va trimite automat mesajul reținut, asigurând continuitatea datelor.

7. **Gestionarea mesajelor**:

   Brokerul gestionează fluxul de mesaje în funcție de abonamentele clienților. Astfel, el decide care clienți trebuie să primească mesajele publicate, în funcție de topicurile la care aceștia sunt abonați. Aceasta asigură o comunicare eficientă și organizată, fără suprasolicitarea rețelei cu mesaje inutile.

# Mecanisme MQTT

Pentru a asigura o comunicare eficientă și fiabilă între dispozitive, MQTT folosește mai multe mecanisme esențiale. Acestea contribuie la gestionarea conexiunilor, la detectarea erorilor și la garantarea livrării mesajelor.

## 1. Keep Alive

**Definiție**: Mecanismul *Keep Alive* asigură că atât clientul, cât și brokerul sunt conștienți că cealaltă parte este încă activă și conectată.

**Funcționare**: Când un client se conectează la broker, poate seta un interval de timp pentru *Keep Alive*, exprimat în secunde. Acesta reprezintă perioada maximă de inactivitate permisă înainte ca o verificare să fie necesară.

**Pachete PING**: Dacă nu au fost trimise alte pachete pe durata intervalului specificat, clientul va trimite un pachet *PINGREQ* pentru a anunța brokerul că este activ. Brokerul va răspunde cu un pachet *PINGRESP* pentru a confirma recepția.

**Deconectare**: În cazul în care brokerul nu primește niciun pachet de la client (inclusiv *PINGREQ*) în intervalul de *Keep Alive* stabilit, acesta va considera clientul deconectat și va închide conexiunea.

## 2. Last Will and Testament (LWT)

**Definiție**: *Last Will and Testament* (LWT) este un mesaj special pe care clientul îl configurează la conectare și care va fi publicat de broker în cazul unei deconectări neașteptate.

**Configurare**: La conectare, clientul poate defini un "mesaj de ultimă voință" (*last will message*), un "topic de ultimă voință" (*last will topic*), un nivel QoS și un *retain flag*. Acest mesaj va fi publicat dacă brokerul detectează o deconectare anormală a clientului.

**Funcționare**: Dacă clientul se deconectează fără a trimite pachetul *DISCONNECT* corespunzător, brokerul va publica *last will message* pe *last will topic* cu setările prestabilite de QoS și *retain flag*.

**Utilitate**: Acest mecanism permite dispozitivelor din rețea să fie notificate atunci când un client se deconectează neintenționat, permițându-le să ia măsuri în consecință (de exemplu, pentru a remedia o problemă sau pentru a ajusta funcționarea altor dispozitive).

### 3. Quality of Service (QoS)

**Definiție**: *Quality of Service* (QoS) este un mecanism care definește nivelul de fiabilitate în livrarea mesajelor între client și broker.

**Nivele de QoS**:

- **QoS 0 – Cel mult o dată**: Mesajul este transmis fără nicio confirmare de la destinatar. Mesajul poate să nu ajungă deloc sau poate fi livrat de mai multe ori în cazul unor probleme de retransmisie.
- **QoS 1 – Cel puțin o dată**: Mesajul este livrat cu confirmare, garantând că destinatarul îl primește cel puțin o dată. Totuși, din cauza mecanismului de confirmare, mesajul poate ajunge de mai multe ori.
- **QoS 2 – Exact o dată**: Aceasta este cel mai înalt nivel de fiabilitate. Se folosește un mecanism de schimb de patru pași pentru a asigura că mesajul este livrat și procesat exact o dată, fără duplicare.

# Implementare

## 1. Structura bazei de date

Vom folosi o bază de date pentru a stoca toate informațiile relevante rulării brocker-ului.

1. **Tabela user_info**

```
CREATE TABLE user_info (
id INT AUTO_INCREMENT PRIMARY KEY,
client_id VARCHAR(255) UNIQUE,
username VARCHAR(255),
login_time DATETIME,
disconnect_time DATETIME,
session_expiry_interval INT,
is_subscriber TINYINT(1) -- 1 for true, 0 for false
);
```

2. **Tabela subscription_info**

```
CREATE TABLE subscription_info (
id INT AUTO_INCREMENT PRIMARY KEY,
subscriber_id INT,
topic VARCHAR(255),
requested_qos TINYINT,
FOREIGN KEY (subscriber_id) REFERENCES user_info(id)
);
```

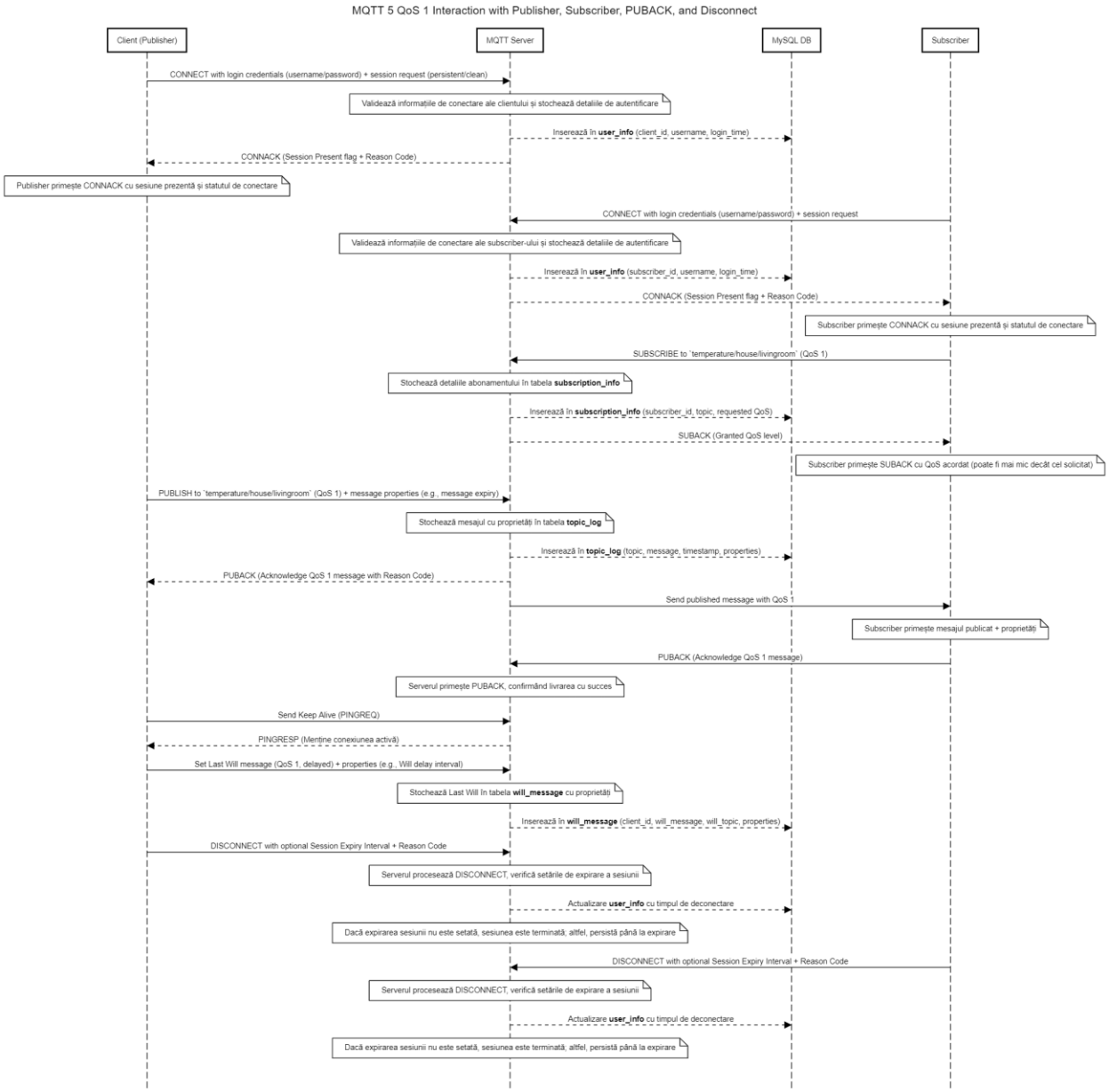
3. **Tabela topic_log**

```
CREATE TABLE topic_log (
id INT AUTO_INCREMENT PRIMARY KEY,
topic VARCHAR(255),
message TEXT,
timestamp DATETIME,
message_expiry INT, -- Expiry time in seconds
qos_level TINYINT, -- QoS level (0, 1, 2)
other_properties VARCHAR(255) -- Other properties as a VARCHAR
);
```

4. **Tabela will_message**

```
CREATE TABLE will_message (
id INT AUTO_INCREMENT PRIMARY KEY,
client_id VARCHAR(255),
will_message TEXT,
will_topic VARCHAR(255),
will_delay_interval INT,
FOREIGN KEY (client_id) REFERENCES user_info(client_id)
);
```

2. Diagrama interacțiunilor




# Bibliografie


https://www.hivemq.com/mqtt/mqtt-5/
https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
https://www.embedded.net.ua/#!mqtt