Etape in realizarea proiectului

MQTT v5 server

- **Înțelegerea noțiunilor de bază ale protocolului MQTT**
  - Specificațiile protocolului MQTT:
    - Am început prin a citi specificațiile MQTT v5. În acest document este definit modul în care ar trebui să se comporte clienții și serverele, inclusiv structurile pachetelor, nivelurile QoS și codurile de eroare.
  - Fluxul de lucru al protocolului:
    - Am analizat fluxul de mesaje și schimburi de pachete într-o sesiune MQTT. Aceasta include: CONNECT, PUBLISH, SUBSCRIBE, DISCONNECT și pachetele de confirmare**.**
- **Proiectarea arhitecturii serverului**
  - Structura modulara
    - Modul de gestionare a conexiunilor:
      - Responsabilități:
        - Gestionează conexiunile primite și gestionează deconectările. Deschide socket-uri TCP pentru a asculta pe portul specificat (de obicei 1883) și procesează fiecare cerere de conectare de la clienți.
      - Caracteristici:
        - Acceptă conexiunile și stochează detaliile clientului (cum ar fi IP, port și informații despre sesiune).
        - Monitorizează intervalele keep-alive și deconectează clienții care depășesc perioada de inactivitate permisă.
        - Inițiază mesaje Last Will dacă un client se deconectează în mod neașteptat.
        - Colaborează îndeaproape cu modulul de autentificare pentru a autoriza clienții în timpul configurării conexiunii.
    - Modulul de gestionare a sesiunilor:
      - Responsabilități:
        - Stochează și recuperează datele de sesiune ale clienților. MQTT v5 necesită ca serverul să gestioneze sesiunile persistente, care păstrează datele în timpul reconectărilor clientului.
      - Caracteristici:
        - Urmărește datele sesiunii, inclusiv abonamentele, stările mesajelor și setările nivelului QoS.
        - Gestionează expirarea sesiunii, ceea ce ajută la eliminarea sesiunilor neutilizate și la conservarea resurselor.

- Asigură stocarea și recuperarea în siguranță a datelor de sesiune pentru clienții cu sesiuni persistente, permițând reconectări fără pierderea stării.

- Modul de rutare a mesajelor și de gestionare QoS:
  - Responsabilități:
    - Rutează mesajele între clienți pe baza subiectelor la care s-au abonat și gestionează nivelurile de calitate a serviciului (QoS).
  - Caracteristici:
    - Procesează și redirecționează pachetele PUBLISH către clienții abonați la subiectele relevante.
    - Implementează toate nivelurile QoS:
      - QoS0: Transmite mesajele fără a solicita confirmare.
      - QoS1: confirmă livrarea mesajului cu PUBACK și retransmite dacă nu se confirmă.
      - QoS2: asigură livrarea mesajelor exact o singură dată printr-un handshake în patru etape (PUBREC, PUBREL, PUBCOMP).
    - Stochează mesajele reținute pentru anumite subiecte, după cum este necesar.
  - Modul de gestionare a subiectelor:
    - Responsabilități:
      - Gestionează subiectele și urmărește ce clienți sunt abonați la fiecare subiect. De asemenea, reține mesajele la cerere.
    - Caracteristici:
      - Menține un registru al tuturor subiectelor active și păstrează o listă a clienților abonați la fiecare subiect.
      - Gestionează abonamentele la subiecte wildcard
      - Suportă mesaje reținute, asigurându-se că ultimul mesaj cunoscut pe un subiect este transmis noilor abonați.
  - Modul de autentificare și autorizare:
    - Responsabilități:
      - Asigură că numai clienții autorizați se pot conecta.
    - Caracteristici:
      - Implementează autentificarea de bază cu nume de utilizator/parolă, cu opțiunea de a se extinde la autentificarea bazată pe certificate.
    - Suportă autorizarea bazată pe roluri prin definirea clienților care pot publica sau se pot abona la anumite subiecte.
  - Modul de logare și gestionare a erorilor:
    - Responsabilități:
      - Înregistrează evenimentele cheie și erorile, furnizând date pentru depanare și monitorizare.
    - Caracteristici:

        - Consemnează evenimentele importante, cum ar fi conexiunile, deconectările, schimburile de mesaje și erorile.
        - Oferă feedback clienților prin intermediul codurilor de eroare conforme cu MQTT v5, ajutându-i să identifice problemele.
        - Stochează jurnalele în mod persistent, dacă este necesar, pentru analiză sau depanare ulterioară.
    - Concurrență și scalabilitate
        - Modelul Thread-per-Connection:
            - Cum funcționează:
                - Fiecare conexiune client primește propriul fir. Acest model este ușor de înțeles și de implementat, dar poate avea limitări în cazul unui număr mare de conexiuni.
            - Avantaje: Simplu și permite fiecărei conexiuni client să funcționeze independent.
            - Dezavantaje: Firele pot fi mari consumatoare de resurse; dacă sute sau mii de clienți se conectează, serverul poate avea probleme cu gestionarea resurselor și cu comutarea contextului firelor.
- **Bazele programării socket**
    - Deoarece vor fi utilizate socket-uri in mod direct, este esențială programarea socket-urilor la nivel scăzut:
        - Configurarea socket-ului:
            - Un server TCP va fi configurat care ascultă pe un port, acceptă conexiuni ale clienților, generand un handler pentru fiecare client și face schimb de date.
- **Implementarea parserului și serializatorului de pachete MQTT**
    - MQTT se bazează pe un protocol binar, deci vor trebui analizate și construite pachete binare în conformitate cu specificațiile MQTT v5:
        - Tipuri de pachete:
            - CONNECT/CONNACK: Gestionează inițierea conexiunii și răspunde cu succes sau eșec.
            - PUBLISH/PUBACK/PUBREC/PUBREL/PUBCOMP: Gestionează publicarea și confirmarea mesajelor pe baza QoS.
            - SUBSCRIBE/SUBACK: Gestionează abonamentele la subiecte.
            - UNSUBSCRIBE/UNSUBACK: Gestionează eliminarea abonamentelor.
            - DISCONNECT: Închiderea sesiunii unui client.
        - Proprietăți și metadate:
            - Gestionarea proprietăților MQTT v5 (de exemplu, intervalul de expirare a sesiunii, intervalul de expirare a mesajului) prin citirea și scrierea acestor câmpuri în pachete.
        - Coduri de eroare:
            - Vom implementa codurile și condițiile de eroare MQTT v5 pentru a face serverul să fie conform cu protocolul.
- **Construirea funcțiilor MQTT de bază pas cu pas**
    - Vom implementa fiecare caracteristică de bază individual:
        - CONNECT și CONNACK:

- Scop: Pachetele CONNECT și CONNACK sunt utilizate pentru inițierea unei sesiuni MQTT. Aceasta este prima interacțiune dintre un client și server și stabilește parametrii sesiunii.
- Etape de implementare:
    - Parsarea pachetului CONNECT:
        - Extragem câmpurile necesare, cum ar fi ID-ul clientului, clean session flag, intervalul keep-alive, datele de autentificare și alte proprietăți ale conexiunii.
    - Validarea parametrilor conexiunii:
        - Verificam ID-ul clientului și validați dacă toate câmpurile necesare sunt prezente.
    - Crearea sau restaurarea sesiunii:
        - Dacă clean session flag este setat la fals, încercăm să restaurăm o sesiune anterioară pentru ID-ul clientului.
        - Dacă clean session flag este true , începem o sesiune nouă, eliminând toate datele sesiunii anterioare.
    - Respond with CONNACK:
        - trimitem un pachet CONNACK către client, indicând succesul sau eșecul. Pachetul CONNACK include:
            - Indicator de sesiune prezentă: Indică dacă a fost restaurată o sesiune anterioară.
            - Codul de retur: Indică starea conexiunii (succes, eșec de autorizare etc.).
    - Initiate Keep-Alive Timer: pornește un cronometru pentru intervalul de menținere în viață, dacă este furnizat de client. Serverul trebuie să deconecteze clienții care depășesc intervalul keep-alive fără a trimite date.
- Gestionarea sesiunii:
    - Scop: sesiunile MQTT permit clienților să se reconecteze și să continue de unde au rămas. Sesiunile stochează informații specifice clientului, cum ar fi abonamentele și mesajele neacordate.
    - Etape de implementare:
        - Stocarea datelor sesiunii:
            - Atunci când se creează o sesiune, se stochează informațiile clientului, inclusiv subiectele la care acesta s-a abonat și cerințele QoS.
        - Sesiuni persistente:
            - Dacă clientul solicită o sesiune persistentă, păstrăm datele în timpul deconectărilor. Stocăm mesajele QoS1 și QoS2 neacceptate pentru a le transmite atunci când clientul se reconectează.
        - Expirarea sesiunii:
            - Pentru gestionarea eficientă a resurselor, implementăm un mecanism de expirare a sesiunii, ștergând sesiunile inactive după un anumit timp.

- Mecanismul Publish/Subscribe:
    - Scop: Modelul publish/subscribe este nucleul comunicării MQTT, permițând clienților să trimită (PUBLISH) și să primească (SUBSCRIBE) mesaje pe teme specifice.
    - Etape de implementare:
        - Tratarea pachetelor PUBLISH:
            - Analizăm pachetul PUBLISH primit pentru a extrage subiectul, sarcina utilă a mesajului și nivelul QoS.
            - Pentru mesajele reținute, se stochează mesajul ca „ultimul mesaj bun cunoscut” pentru subiect.
            - Pentru QoS1 și QoS2, stocăm informații privind starea mesajului pentru confirmare și retransmisie.
        - Livrarea mesajelor către abonați:
            - Verificăm subiectul mesajului PUBLISH și identifică. toți clienții abonați la subiectul respectiv.
            - Transmitem mesajul în funcție de nivelul de abonare QoS al fiecărui client.
        - Tratarea pachetelor SUBSCRIBE:
            - Parsăm pachetul SUBSCRIBE pentru a extrage subiectele și nivelurile QoS.
            - Adăugăm clientul pe lista de abonați pentru fiecare subiect, respectând nivelul QoS.
            - Trimitem un pachet SUBACK clientului pentru a confirma statutul de abonat.
        - Tratarea pachetelor UNSUBSCRIBE:
            - Scoatem clientul din lista de abonați pentru fiecare subiect specificat.
            - Trimitem un pachet UNSUBACK pentru a confirma succesul dezabonării.

- .Niveluri QoS: Implementați diferitele niveluri QoS:
    - QoS0 (cel mult o dată)
        - Scop: Livrează mesaje fără a solicita confirmare de primire, ceea ce înseamnă că mesajele sunt trimise o singură dată și nu sunt stocate sau retrimise de server.
        - Implementare:
            - La primirea unui pachet PUBLISH cu QoS0, serverul direcționează mesajul către clienții abonați fără a aștepta nicio confirmare de primire.
            - Nu este necesar să se stocheze starea mesajului sau să se retransmită în caz de eșec.

- QoS1 (cel puțin o dată)
    - Scop: garantează că mesajele sunt livrate cel puțin o dată, necesitând confirmarea de primire din partea clientului care le primește.
    - Implementare:
        - Stocarea stării mesajului: Stocăm fiecare mesaj PUBLISH cu QoS1 până când este confirmat de client.
        - Trimiterea pachetului PUBLISH: Transmitem mesajul către clienții cu abonamente QoS1 și așteaptăm un răspuns PUBACK.
        - Handle PUBACK: la primirea unui PUBACK de la client, marcăm mesajul ca fiind confirmat și îl eliminăm din coada de retransmisie.
        - Retransmitere: Dacă PUBACK nu este primit într-un interval de timp specificat, retransmitem mesajul.
- QoS2 (Exact o dată)
    - Scop: garantează livrarea exact o singură dată, folosind un handshake în patru pași pentru a preveni mesajele duplicate.
    - Implementare:
        - Pachetul PUBLISH inițial: Trimitem pachetul PUBLISH către clienții abonați cu QoS2.
        - Primirea PUBREC: clientul răspunde cu un pachet PUBREC pentru a confirma primirea. În acest moment, serverul consideră mesajul „în curs de desfășurare”.
        - Trimite PUBREL: Serverul răspunde la PUBREC cu un pachet PUBREL, semnalând că așteaptă ca clientul să confirme finalizarea.
        - Primește PUBCOMP: La primirea PUBCOMP de la client, mesajul este complet confirmat, iar serverul îl poate elimina din stocare.

- Mecanismul keep-alive
    - Scop: Mecanismul keep-alive asigură că clienții rămân receptivi. Dacă un client nu comunică în intervalul keep-alive, serverul încheie conexiunea.
    - Implementare:
        - Monitorizarea intervalului Keep-Alive: Pentru fiecare client conectat, este urmărită ultima oră de comunicare.
        - Aplicarea limitelor de păstrare a vieții: Dacă un client depășește intervalul keep-alive fără să trimită date, se presupune că s-a deconectat și închideți conexiunea.
- Last Will
    - Scop: mesajul Last Will permite clienților să specifice un mesaj final care să fie trimis abonaților în cazul în care se deconectează în mod neașteptat.
    - Implementare:

        - Setăm mesajul Last Will la CONNECT: În timpul conectării, clientul își specifică mesajul Last Will, inclusiv subiectul, mesajul și nivelul QoS.
        - Stocarea mesajului Last Will: Păstreăm mesajul Last Will ca parte a stării de sesiune a clientului.
        - Trimitem mesajul Last Will la deconectarea neașteptată: Dacă clientul se deconectează în mod neașteptat (fără a trimite un pachet DISCONNECT), publicăm mesajul Last Will la subiectul specificat cu nivelul QoS definit.

- **Resurse utilizate**
    - **MQTT v5 Specifications**: Documentația oficială MQTT v5.0 oferă detalii complete privind structura și comportamentul protocolului.
    - **Socket Programming Guides**:
        - **C**: GeeksforGeeks oferă un tutorial informativ privind programarea socket-urilor în C, acoperind atât implementările client, cât și cele server. GeeksforGeeks
    - **MQTT Protocol and Libraries**:
        - **Mosquitto**: Un broker MQTT open-source care poate servi drept referință pentru cele mai bune practici în implementarea serverului MQTT. Codul sursă este disponibil pe GitHub. Steve's Internet Guide