Documentație pentru serverul MQTT, implementat la disciplina RC-P

Cebotaroș Emil

Chiriac Ion

Munteanu Letiția-Ioana

grupa 1307A, Facultatea de Automatică și Calculatoare
Universitatea Tehnică "Gheorghe Asachi", Iași, România

January 19, 2021

Cuprins

Capitolul 1

Introducere

MQTT este un protocol de comunicare M2M (Machine 2 Machine), utilizat frecvent în aplicații IoT. Aplicația noastră implementează versiunea 5 a protocolului.

1.1 Documentație

În cadrul etapei de proiectare, au fost identificate 22 clase necesare pentru a asigura ușurința procesului de dezvoltare și modularitatea aplicației.

1.1.1 Clase de date

Clasa BinaryData

MQTT codifică anumite date de tip binar într-o formă ușoară pentru procesare. Respectiv, această clasă va aveam câmpul `length` de tip int, și `data` de tip `bytes`. Din metodele implementate, avem o metodă statică `fromBytesToBinary`, care realizează operația de conversie din `bytes` în date de tip `BinaryData`. Fiecare obiect de tip `BinaryData` are 2 metode proprii, `getLength`, care returnează lungimea conform protocolului și `getData`, care returnează octeții sub forma de `BinaryData`.

VariableByte

MQTT codifică anumite date de tip întreg într-un număr variabil de octeți. Respectiv, această clasă va avea câmpul `data`, de tip `bytes`. Metodele statice implementate de această clasă vor fi : `encode` și `decode`. Prima se utilizează pentru a transforma dintr-un număr întreg într-un obiect de tip `Variable Byte`, a doua se utilizează pentru a transforma un obiect de tip `VariableByte` într-un întreg.

CustomUTF8

MQTT codifică anumite date de tip string în format Unicode. Python3 consideră în mod implicit obiectele de tip string ca fiind în format Unicode. Totuși, formatarea Python nu corespunde cu specificațiile implementării MQTT. Din această cauză este necesară implementarea clasei CustomUTF8. Aceasta va avea câmpul `data`, de tip `bytes`. Metodele implementate de această clasă vor fi statice : `encode` și `decode`. Prima metodă convertește un obiect de tip `string` într-un obiect de tip `CustomUTF8`. A doua metodă convertește un obiect de tip `CustomUTF8` într-un obiect de tip `string`.

1.1.2 Configurare

Pentru configurarea serverului, am decis să avem un fișier de configurare *config/srv.conf*, asemănător după structură cu fișierul de configurare pentru serviciul SSH. Acesta va conține setările pe mediul de stocare intern al gazdei pe care rulează brokerul. În repozitoriul de Github există fișierul *config/srv.conf.bak*.
Pentru a asigura că aceste setări sunt mereu actuale, a fost creată clasa Config.
Câmpurile acesteia sunt: `AllowPublicAccess`,`AuthenticationMethods`,`KnownClientsPath`, `HistoryMessageQueueSize`,`RetainedMessagesPath`, `DefaultKeepAlive`, `TopicHierarchyListPath`, `MaxPacketSize`, `MaxQoS1QueueSize`, `MaxQoS2QueueSize`, `MaxSupportedTopics`, `SessionsPath`, `AllowWildcard`, `configPath`.
Pe lângă constructorul suprascris, clasa pune la dispoziție metoda statică `parseConfString`, care mapează valorile literale din fișierul de configurare în tipuri de bază ale limbajului Python. Aceasta este apelată în metoda `reload`, care încarcă datele din fișier și mapează configurările specificate în fișier asupra configurărilor în Python. Pe lângă acestea, clasa mai conține metoda `getKnownTopics`, care ar trebui apelată după `reload`. Aceasta returnează lista de topicuri cunoscută de serverul MQTT.

1.1.3 Clasa MQTTPacket

Deoarece MQTT este un protocol bazat pe pachete, am considerat ca fiind imperativă crearea unei clase generice care să modeleze trăsăturile comune ale unui pachet MQTT. Această clasă câmpuri de bază, având câte o metodă de parsare pentru fiecare.

Implementare generală în Python Clasa generică `MQTTPacket` utilizează câmpurile `fixed`, `variable` și `payload` pentru a menține datele referitoare la conținutul pachetelor.
Metodele de parsare vor adăuga câmpuri noi în interiorul câmpurilor. Acestea vor fi inserate înăuntrul dicționarelor. Aceste metode sunt :
`parseFixedHeader`, `parseVariableHeader`, `parsePayloadHeader`. Pe lângă aceste metode de bază, există și metoda `parse`, care va fi suprascrisă în fiecare subclasă a clasei `MQTTPacket`.

1.1.4 Subclasele MQTTPacket

Clasa ConnectPacket

Clasa ConnectPacket descrie un pachet MQTT de tip CONNECT. Aceasta va moșteni din clasa generică `MQTTPacket` și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader`, `parsePayloadHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin datele parsate în format human-readable.

Clasa ConnackPacket

Clasa ConnackPacket descrie un pachet MQTT de tip CONNACK. Aceasta va moșteni din clasa generică MQTTPacket, pentru a menține legătura logică între pachete. Totuși, metodele moștenite nu vor fi suprascrise, ci va fi adăugată o metodă statică `generatePacketData`, care va genera conținutul pachetului de tip CONNACK.

Clasa PublishPacket

Clasa PublishPacket descrie un pachet MQTT de tip PUBLISH. Aceasta va moșteni din clasa generică `MQTTPacket` și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader`, `parsePayloadHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin datele parsate în format human-readable.

Clasa PubackPacket

Clasa PubackPacket descrie un pachet MQTT de tip PUBACK. Aceasta va moșteni din clasa generică MQTTPacket, pentru a menține legătura logică între pachete. Totuși, metodele moștenite nu vor fi suprascrise, ci va fi adăugată o metodă statică `generatePacketData`, care va genera conținutul pachetului de tip PUBACK.

Clasa PubrecPacket

Clasa PubrecPacket descrie un pachet MQTT de tip PUBREC. Aceasta va moșteni din clasa generică MQTTPacket și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader` și `parse`. În plus, aceasta are și metoda statică `generatePacketData`, care va genera pachetul pentru a fi trimis în rețea.

Clasa PubrelPacket

Clasa PubrelPacket descrie un pachet MQTT de tip PUBREL. Aceasta va moșteni din clasa generică MQTTPacket și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader` și `parse`. În plus, aceasta are și metoda statică `generatePacketData`, care va genera pachetul pentru a fi trimis în rețea.

Clasa PubcompPacket

Clasa PubcompPacket descrie un pachet MQTT de tip PUBCOMP. Aceasta va moșteni din clasa generică `MQTTPacket` și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin datele parsate în format human-readable. Metoda `parsePayloadHeader` nu va avea nevoie de suprascriere pentru că în specificație este menționat faptul că pachetele de tip PUBCOMP nu au payload.

Clasa SubscribePacket

Clasa SubscribePacket descrie un pachet MQTT de tip SUBSCRIBE. Aceasta va moșteni din clasa generică `MQTTPacket` și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader`, `parsePayloadHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin datele parsate în format human-readable.

Clasa SubackPacket

Clasa SubackPacket descrie un pachet MQTT de tip SUBACK. Aceasta va moșteni din clasa generică `MQTTPacket` și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader`, `parsePayloadHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin datele parsate în format human-readable. În plus, aceasta are și metoda statică `generatePacketData`, care va genera pachetul pentru a fi trimis în rețea.

Clasa UnsubscribePacket

Clasa UnsubscribePacket descrie un pachet MQTT de tip UNSUBSCRIBE. Aceasta va moșteni din clasa generică `MQTTPacket` și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader`, `parsePayloadHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin datele parsate în format human-readable.

Clasa UnsubackPacket

Clasa UnsubackPacket descrie un pachet MQTT de tip UNSUBACK. Aceasta va moșteni din clasa generică `MQTTPacket` și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader`, `parsePayloadHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin datele parsate în format human-readable. În plus, aceasta are și metoda statică `generatePacketData`, care va genera pachetul pentru a fi trimis în rețea.

Clasa PingreqPacket

Clasa PingreqPacket descrie un pachet MQTT de tip PINGREQ. Aceasta va moșteni din clasa generică `MQTTPacket` și vor suprascrie metodele `parseFixedHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin

datele parsate în format human-readable. Metodele `parseVariableHeader` și `parsePayloadHeader` nu vor avea nevoie de suprascriere pentru că în specificație este menționat faptul că pachetele de tip PINGREQ nu au nici header variabil, nici payload.

Clasa PingrespPacket

Clasa PingrespPacket descrie un pachet MQTT de tip PINGRESP. Aceasta va moșteni din clasa generică `MQTTPacket` și vor suprascrie metodele `parseFixedHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin datele parsate în format human-readable. Metodele `parseVariableHeader` și `parsePayloadHeader` nu vor avea nevoie de suprascriere pentru că în specificație este menționat faptul că pachetele de tip PINGRESP nu au nici header variabil, nici payload.

Clasa AuthPacket

Clasa AuthPacket descrie un pachet MQTT de tip AUTH. Aceasta va moșteni din clasa generică `MQTTPacket` și va suprascrie metodele `parseFixedHeader`, `parseVariableHeader` și `parse`. În urma execuției metodelor, vor fi generate dicționare care conțin datele parsate în format human-readable. Metoda `parsePayloadHeader` nu va avea nevoie de suprascriere pentru că în specificație este menționat faptul că pachetele de tip AUTH nu au payload.

1.1.5 Clasa Watchdog

Clasa Watchdog a fost introdusă pentru a realiza operații care presupun accesul la toate threadurile clienților. Din această cauză, aceste threaduri sunt salvate în câmpul `threads`. Operațiile asupra threadurilor sunt implementate prin intermediul metodelor `purgeThreadBySocket`, care deconectează forțat un client, închizându-i socketul de comunicare. Un analog al acesteia este metoda `forceDisconnectByClientId`, care deconectează forțat un client după id-ul de client utilizat. Pentru a găsi rapid lista id-urilor utilizate de clienți, se folosește metoda `getUsedClientIDs`. Metoda `getSubscriberSockets` se utilizează pentru a facilita transmiterea mesajelor de tip PUBLISH către clienții abonați la topic. Primind în calitate de parametri topicul propriu-zis și socketul clientului care a trimis pachetul PUBLISH, aceasta va returna o listă de socketuri în care să fie scrise pachetele de tip PUBLISH. Metoda care verifică unicitatea id-urilor conectate este `isUsedClientID`, care în spate apelează metoda `getUsedClientIDs`. Pentru implementarea funcționalității de broadcast, au fost realizate metodele `broadcast` și `broadcastByTopic`.

1.1.6 Clasa Authenticator

În cazul în care nu folosim pachetul AUTH, protocolul MQTT permite logarea prin intermediul specificării unui username și a unei parole. Clasa Authenticator este specializată în implementarea părții de User Access Control. Aceasta

conține câmpurile `config`, și `clients`. Câmpul `config` stochează obiectul care conține configurările, iar câmpul `clients` stochează credențialele cunoscute, parsate din fișierul specificat în `config`. Această clasă implementează 2 metode : una normală, și una statică. Cea normală este metoda `getKnownCreds`, care extrage din fișierul specificat în fișierul de configurare combinațiile cunoscute de utilizatori. Cea statică este metoda `authenticate`, care face verificarea dacă credențialele sunt acceptate și cunoscute de către server.

1.1.7 Clasa Session

Clasa Session a fost inclusă în design pentru a stoca datele care țin de o sesiune de comunicare client-broker. Aceasta include și metode pentru comunicarea cu clienții, gestionând automat generarea de pachete de răspuns prin intermediul metodei `handlePacket`.

Structură generală în Python Clasa conține următoarele câmpuri : `data`, `topics`, `will`, `clientID`, `config`, `auth`, `watchdog`, `socket`. Aceste câmpuri conțin cele mai recente date sub formă de octeți, lista de topicuri la care este abonat clientul, dicționarul care conține informațiile referitoare la *Last Will*, ID-ul clientului, obiectul care conține configurările, obiectul realizează verificarea autentificării, obiectul care conține restul clienților și socketul clientului. Metodele apelate regulat sunt `registerNewData`, ce resetează datele din sesiune legate de ultimul pachet sosit, `classifyData`, ce grupează datele într-un pachet, împreună cu tipul acestuia, returnând o subclasă a `MQTTPacket`, și `handleConnection`, a cărei funcționalitate depinde mult de tipul pachetului de intrare. În cazul în care la conectare este setat bitul `cleanStart`, se apelează metoda `reset`, care uită toate informațiile din sesiunea anterioară.