Documentatie Server CoAP- Remote Storage

1. Introducere

Scopul acestui proiect este implementarea unui server CoAP (Constrained Application Protocol) care permite stocarea și gestionarea fișierelor trimise de un client într-o arhitectură de tip remote storage.
Protocolul CoAP este conceput pentru sisteme cu resurse limitate (dispozitive IoT, senzori, microcontrolere), fiind o alternativă ușoară la HTTP, bazată pe UDP.
Serverul implementat oferă o serie de funcționalități:

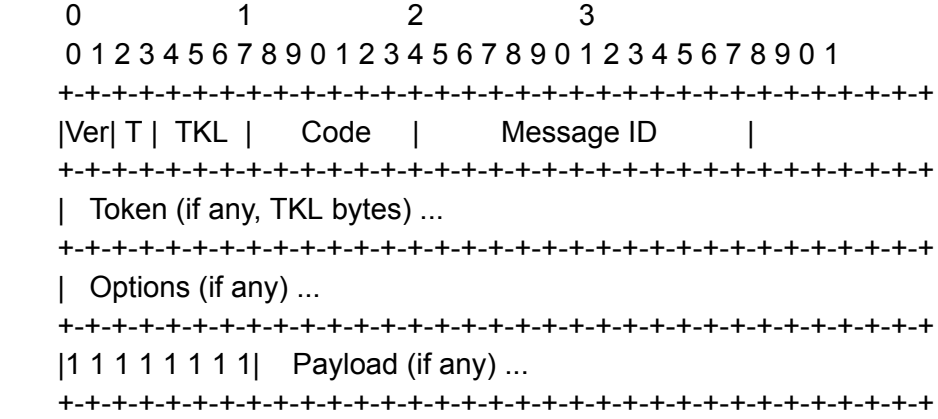
-încărcarea și descărcarea fișierelor
-crearea și ștergerea fișierelor
-navigarea în structura de directoare
-mutarea fișierelor între directoare

Comunicarea între client și server se face exclusiv prin mesaje CoAP, transmise prin socket-uri UDP.

2.Formatul pachetelor [1]

Un pachet CoAP reprezintă unitatea de bază de comunicare între client și server. Fiecare pachet conține informațiile necesare pentru identificarea, procesarea și livrarea corectă a mesajului. Structura unui pachet CoAP este alcătuită din mai multe câmpuri, fiecare având un rol bine definit.

Structura pachetelor este următoarea:



-Header: are o lungime fixă de 4 octeți și este poziționat la începutul fiecărui pachet. Acesta conține informații generale despre mesaj, cum ar fi versiunea protocolului, tipul mesajului(Confirmable (0), Non-confirmable (1), Acknowledgement (2), Reset (3)), codul metodei (GET, POST, etc.) și un identificator unic de mesaj (Message ID).

-Token; asigură corelarea cererilor cu răspunsurile. Clientul generează o cerere către server, incluzând un token unic care se va regăsi în răspunsul trimis. Utilizarea acestuia este opțională.

-Options: conțin informații suplimentare despre mesaj, cum ar fi tipul de conținut, calea resursei, dimensiunea datelor sau parametri specifici aplicației.

-Payload; reprezintă datele transmise între client și server. Poate conține, de exemplu, conținutul unui fișier, structura unui director sau un mesaj de confirmare.

Dimensiunea maximă recomandată pentru un pachet CoAP este de aproximativ 1152 octeți.Din acest total, payloadul poate ocupa de regulă până la 1024 octeți.Dacă mesajul depășește această limită, conținutul trebuie împărțit în mai multe pachete. Fragmentarea se realizează la nivelul aplicației, iar fiecare pachet trebuie identificat astfel încât receptorul să poată reconstrui corect mesajul complet.

Tipuri de mesaje: [2]

Protocolul CoAP definește patru tipuri principale de mesaje, indicate în câmpul Type din Header. Acestea stabilesc modul de confirmare și comportamentul comunicării între client și server:

-CON (Confirmable) – mesaj confirmabil care necesită primirea unui răspuns de tip ACK. Dacă răspunsul nu este primit într-un anumit interval de timp, mesajul este retransmis.

-NON (Non-confirmable) – nu necesită răspuns de tip ACK; este folosit pentru transmisii rapide, unde pierderea ocazională de pachete este acceptabilă.

-ACK (Acknowledgement) – mesaj de confirmare trimis ca răspuns la un mesaj CON, pentru a semnala primirea acestuia.

-RST (Reset) – mesaj de resetare, transmis atunci când un nod primește un pachet pe care nu îl recunoaște sau nu îl poate procesa.

Metode de cerere: [3] [6]

Protocolul CoAP folosește un set de metode similare cu cele din HTTP, care definesc acțiunile efectuate asupra resurselor de pe server:

-GET(cod 0.01): solicită accesul la o resursă de pe server și returnează conținutul acesteia. Ca urmare,răspunsul trebuie să conțină codul 2.05 (Content) pentru a transmite conținutul fișierului sau al directorului solicitat de client.

-POST(cod 0.02): trimite date către server pentru a crea sau actualiza o resursă. În cazul unei cereri POST, răspunsul trebuie să conțină codul 2.01 (Created) dacă fișierul a fost încărcat cu succes pe server sau codul 2.05 (Content) dacă se returnează o confirmare cu metadatele fișierului.

-PUT(cod 0.03): înlocuiește complet conținutul unei resurse existente (opțional, în funcție de implementare). În cazul unei astfel de cereri,răspunsul trebuie să conțină codul 2.04 (Changed) pentru a confirma că fișierul existent a fost suprascris cu succes.

-DELETE(cod 0.04): șterge o resursă de pe server.În cazul unei cereri DELETE, răspunsul trebuie să conțină codul 2.02 (Deleted) pentru a confirma că fișierul sau directorul a fost șters cu succes de pe server.

-În contextul proiectului, va fi definită și o metodă suplimentară mai precis metoda MOVE(cod 0.05), care va fi utilizată pentru mutarea unui fișier anume într-un alt director *existent*, iar răspunsul corespunzător unei astfel de cereri ar avea codul 2.04 (Changed).

Tipuri de răspunsuri: [4] [7]

Răspunsurile CoAP sunt mesaje trimise de server către client pentru a indica rezultatul unei cereri. Ele folosesc câmpul Code din antet pentru a comunica dacă cererea a fost procesată cu succes sau dacă a apărut o eroare.
În aplicația de tip remote storage, răspunsurile confirmă operațiile efectuate asupra fișierelor și directoarelor, precum descărcarea, încărcarea, ștergerea sau mutarea acestora:

-2.05 (Content) este folosit pentru a transmite conținutul fișierelor sau al directoarelor. În cazul unei cereri GET, acest cod indică faptul că datele returnate sunt valide și actuale.

-2.03 (Valid) indică faptul că resursa nu s-a modificat de la ultima interogare.

-2.01 (Created) confirmă că o resursă a fost creată cu succes pe server.

-2.04 (Changed) confirmă că resursa existentă a fost modificată.

-2.02 (Deleted) confirmă ștergerea unei resurse de pe server.

Mecanismul Piggybacked Response: [5]

În cazul mesajelor Confirmable, serverul poate trimite răspunsul în două moduri:

-Piggybacked Response presupune trimiterea directă a răspunsului în același pachet cu mesajul de confirmare (ACK).

-Separate Response determină faptul că serverul trimite mai întâi un mesaj ACK pentru confirmarea cererii și apoi, într-un pachet separat se trimite răspunsul efectiv.

Pachete proprietare (pentru aplicație)

Tip	Descriere	Conținut Payload
FILE_UPLOAD	Client → Server	Header CoAP (POST) + nume fișier + conținut fișier
FILE_DOWNLOAD	Client → Server	Header CoAP (GET) + calea fișierului
FILE_DELETE	Client → Server	Header CoAP (DELETE) + calea fișierului
FILE_MOVE	Client → Server	Header CoAP (MOVE) + calea sursă + calea destinație
DIR_LIST	Client → Server	Header CoAP (GET) + calea directorului

RESPONSE_OK / ERROR	Server → Client	Header CoAP (ACK) + cod + mesaj textual

3.**Interacțiunile client–server**

-Upload fișier (POST /upload)

Client trimite pachet Confirmable cu cod 0.02 (POST)

Payload: [path] + [file content]

Server salvează fișierul → trimite ACK 2.01 Created

-Download fișier (GET /download)

Client trimite GET cu calea fișierului

Server trimite ACK 2.05 Content + payload cu fișierul

-Ștergere fișier (DELETE /path)

Client → Confirmable cod 0.04 (DELETE)

Server → ACK 2.02 Deleted

-Mutare fișier (MOVE /src /dst)

Client → Confirmable cod 0.08 (MOVE)

Server → ACK 2.01 Created dacă mutarea a reușit

-Listare directoare (GET /list)

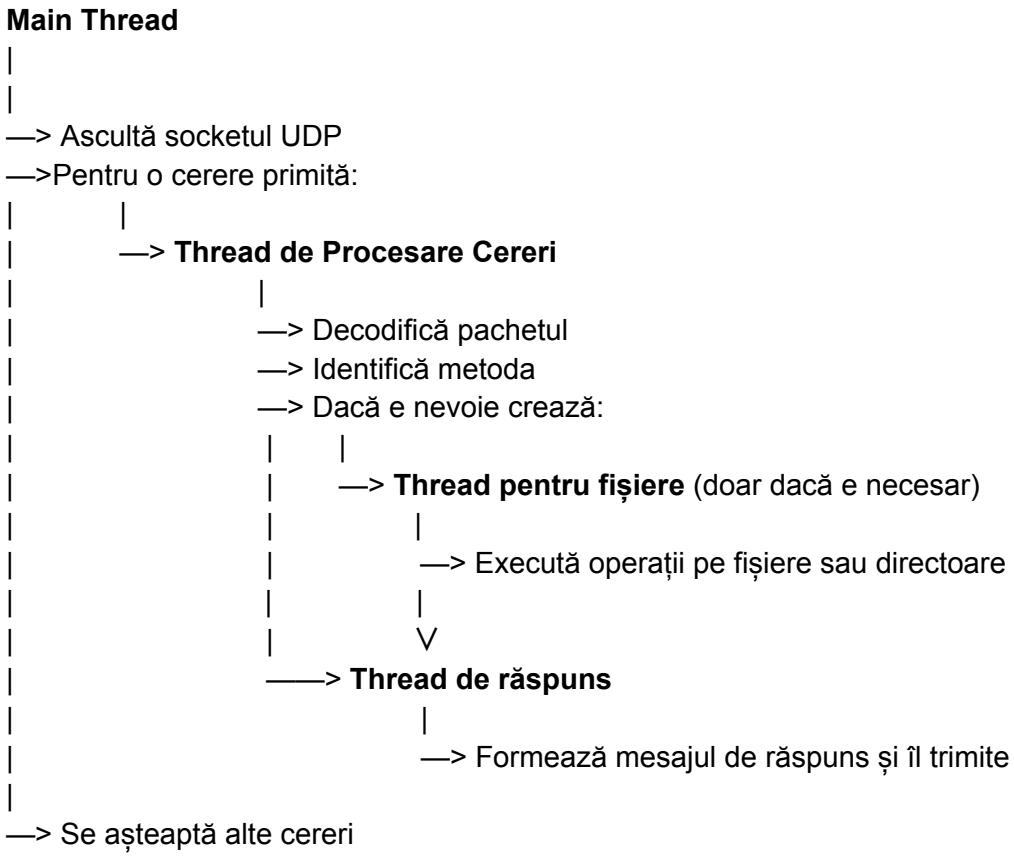
Client → GET pe director

Server → 2.05 Content cu payload (nume fișiere/directoare)

4. **Threading și paralelizare**

Fir	Responsabilitate	Detalii
Thread Main	Ascultă pachete UDP	Socket.recvfrom() într-un loop infinit
Thread de procesare cereri	Decodifică pachetul și identifică metoda	Creează un task pentru fiecare cerere nouă
Thread pentru fișiere	Operații I/O (citire/scriere/mutare)	Evită blocarea firului principal
Thread de răspuns	Trimite pachetele ACK/Response	Confirmă cererea, trimite payload-ul

5. **Schelet logic**



[1] https://datatracker.ietf.org/doc/html/rfc7252#section-3
[2] https://datatracker.ietf.org/doc/html/rfc7252#section-4
[3] https://datatracker.ietf.org/doc/html/rfc7252#section-5.1
[4] https://datatracker.ietf.org/doc/html/rfc7252#section-5.2
[5] https://datatracker.ietf.org/doc/html/rfc7252#section-5.2.1
[6] https://datatracker.ietf.org/doc/html/rfc7252#section-5.8
[7] https://datatracker.ietf.org/doc/html/rfc7252#section-5.9