UNIVERSITATE TEHNICA "GHEORGHE ASACHI" IASI
FACULTATEA DE AUTOMATICA SI CALCULATOARE
SPECIALIZAREL CALCULATOARE SI TEHNOLOGIA INFORMATIEI

Aplicatie de tip browser FS (Client CoAP)

Coordonator:

Prof. Nicolae-Alexandru Botezatu

Studenti:

Curpăn Robert – Gabriel grupa: 1307A
Istrate Sebastian grupa: 1307A

Cuprins:

1.1 Introducere

CoAP (Constrained Application Protocol) este un protocol WEB specializat de transfer, folosit pentru noduri si retele low-power, cu resurse limitate.

Aplicatia cea mai comuna a acestui protocol este domeniul IoT (Internet of Things), in care nodurile "constrained" reprezinta microcontrollere cu putina memorie RAM/ROM, iar retelele low-power se refera la retele personale wireless de putere/viteza scazuta (6LoWPAN = IPv6 over Low-Power Wireless Personal Area Networks.

1.2 Istoric

Standardizarea protocului CoAP a fost facuta de catre un grup numit IETF CoRE(Internet Engineering Task Force, Constrained RESTful Environments), in scopul realizarii comunicarii de tip M2M (Machine To Machine) pentru dispositive electronice de cost redus. IETF este o comunitate deschisa ce are ca punct de interes evolutia arhitecturii Internet si functionarea coerenta a Internetului. Astfel, in ziua de astazi CoAP a ajuns sa fie folosit pentru crearea si manipularea resurselor dintre device-uri, publicarea si subscriptia datelor, accesarea informatiilor despre dispozitivul conectat. Aceste resurse pot fi accesate folosind metode de tip GET si POST.

1.3 Descrierea protocolului CoAP

Modelul de interactiune al CoAP este similar cu modelul client-server al protocolului HTTP. Practic, CoAP foloseste doua tipuri de mesaje:

- Cereri (requests)
- Raspunsuri (responses)

Un request CoAP este trimis de un client pentru a accesa/prelucra o resursa aflata pe un server, actiunea propriu-zisa fiind specificata prin intermediul unui "method code". De cealalta parte, serverul va trimite un raspuns clientului cu ajutorul unui "response code".

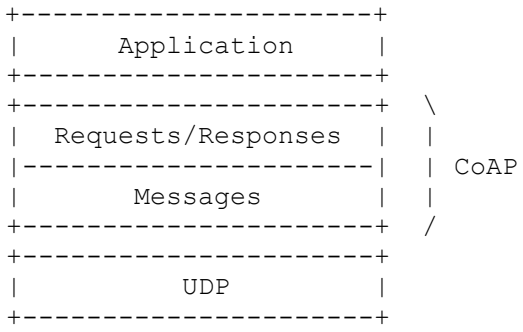

Fig. 1 - Structura CoAP

Spre deosebire de HTTP, CoAP se ocupa cu transferul cereri-raspunsuri intr-un mod asincron, folosind UDP (User Datagram Protocol). Datagramele UDP pot fi trimise catre destinatar, dar nu exista garantia ca aceastea ajung in maniera corecta (mesajele pot avea intarzieri sau pot sa nu ajunga deloc). Pentru a rezolva aceasta problema, protocolul CoAP pune la dispozitie 4 tipuri de mesaje (Confirmable, Non-confirmable, Acknowledgement si Reset) care vor fi detaliate mai jos.

Formatul mesajelor

Inainte de trimiterea unei cereri sau a unui raspuns, mesajului i se ataseaza un header specific protocolului CoAP. Acest header este format din minim 4 octeti si are urmatoarea structura:

CoAP Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
4	32	VER		Type		Token Length				Request/Response Code								Message ID															
8	64	Token (0 - 8 bytes)																															
12	96																																
16	128	Options (If Available)																															
20	160	1	1	1	1	1	1	1	1	Payload (If Available)																							

Fig. 2 – Headerul CoAP

Descrierea campurilor:

- Version -> indica versiunea de CoAP. Aceasta trebuie sa fie 01 din cauza implementarii actuale, altfel se va considera eroare. Celelalte versiuni sunt rezervate pentru urmatoarele implementari de CoAP.

- Type -> sugereaza tipul mesajului
  Request:
  - 00 : CON (confirmable – acest tip de mesaj asteapta un acknowledgement)
  - 01 : NON (non-confirmable – nu asteapta confirmare din partea serverului)

  Response:
  - 10 : ACK (acknowledgement – confirmarea primirii unui request)
  - 11 : RES (reset – indica faptul ca request-ul a fost primit, dar nu a putut fi procesat)

- Token Length -> indica lungimea, in octeti, a campului token. Acest camp este format din 4 biti si poate lua valoarea maxima de 8 , deci campul Token va avea maxim 8 octeti. Pe baza acestui numar se va aloca spatiul corespunzator token-ului. Lungimile din intervalul 9-15 sunt rezervate, iar programul genereaza o eroare in cazul incalcarii acestei reguli.

- Request/Response Code (1 octet) -> message/response codes. Aceste coduri reprezinta tipul cererii pe care vrem sa o trimitem.

- Message ID -> un numar pe 2 octeti ce retine ID-ul mesajului trimis. ID-ul va fi incrementat automat la generarea altor mesaje de tip request. Acest camp ajuta la mentinerea unui "dialog" coerent intre server si client, deoarece serverul ne va raspunde la mesaje cu aceleasi Message ID-uri pentru a sti perechile de mesaje.
- Token -> reprezinta un camp optional al carui dimensiune este dat de *Token Length,* dimensiune generata de client. Token-ul e folosit pentru a potrivi raspunsurile si cererile, indiferent de mesajul continut. El confera securitate comunicarii, deoarece serverul trebuie sa trimita un message-response cu acelasi token ca si cel trimis de client, pentru buna o buna functionare.
- Options -> reprezinta un camp pentru optiuni pe 8 biti, putand fi interpretat in diverse moduri, in functie de valorile celorlalte campuri.
- Payload -> reprezinta un camp de lungime variabila (cu dimensiune maxima de 64KB), separat de header prin intermediul unui octet cu biti setati pe 1 logic. Acest camp va contine informatiile concrete ale cererii, retinute sub forma unui JSON cu 5 campuri.

Codul unei cereri/raspuns (Method Code)

0	1	2	3	4	5	6	7
Class			Code				

Un camp alcatuit din 8 biti. Cei mai semnificativi 3 biti formeaza un numar intitulat "Class", concept asemanator codurilor de status ale protocolului HTTP. Ultimii 5 biti din reprezentarea cererii sau a raspunsului reprezinta un cod care comunica mai multe detalii despre tipul de mesaj. Practic, mesajul poate fi comunicat sub forma "class.code" (c.dd, unde c si d reprezinta cifre zecimale).
Astfel, "class" poate lua valori in intervalul 0-7, iar "code" in intervalul 0-31.
Exemple:

Class = 0.xx – request;
Class = 2.xx – success response;
Class = 4.xx – client error response;
Class = 5.xx – server error response;

Toate celelalte numere asignate clasei sunt rezervate. Un cod special este 0.00 care indica un mesaj gol.

In functie de comenzile implementate, s-au completat method code-urile. Cererile vor avea class = 0, iar code va lua valori astfel incat cererile sa fie de tip: EMPTY, RUN, GET, POST, DELETE.

Pentru a beneficia de intreaga functionalitate a programului, s-au alocat doua method code-uri speciale pentru comenzile register (0.22), respective message (0.23).

Modul de transmitere a mesajelor

Dupa cum am amintit si anterior, protocolul CoAP defineste 4 tipuri de mesaje: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK) si Reset (RES). CON si ACK stau la baza schimbului de mesaje intr-o comunicatie sigura (de incredere), in timp ce NON se foloseste pentru comunicatiile non-reliable. Modul de operare a acestor mesaje este descris in schemele urmatoare:

Mesajele de tip CON (Confirmable):

Clientul trimite o cerere de tip confirmable, iar apoi asteapta 2 mesaje de la server. Primul mesaj va fi un mesaj gol (ce contine doar headerul), cu campul Type setat pe "10", adica ACK. Acest mesaj aduce la cunostinta clientului ca serverul a primit cererea si nu mai este nevoit sa retrimita mesajul. In cazul in care serverul nu are viteza de raspuns suficient de mare la cereri, atunci acest mesaj gol va fi benefic pentru a evita traficul redundant in retea. Dupa, clientul asteapta cel de-al doilea mesaj, care contine mesajul propriu zis cu raspunsul la cererea facuta. Acest mesaj este de tip Confirmable, deci serverul ne cere sa ii confirmam ca am primit ultimul mesaj, altfel il va trimite din nou, pana cand se asigura ca a ajuns la destinatie. Clientul va trebui sa trimita un mesaj gol de tip "ACK" pentru a confirma cele enuntate anterior.

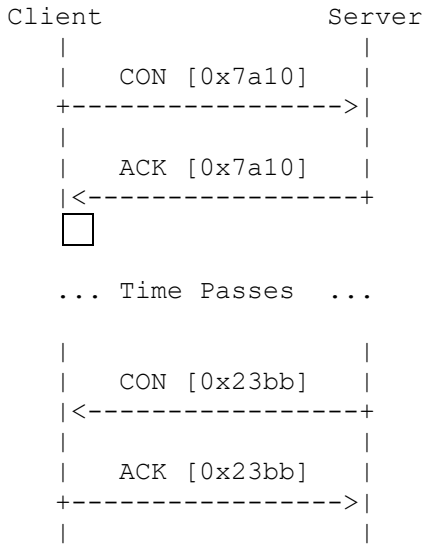

Figura 3 – Transmiterea mesajelor in cadrul CoAP (CON)

Mesajele de tip NON (Nonconfirmable):

Clientul trimite o cerere catre server, apoi asteapta un singur raspuns ce va contine tot mesajul dorit. Aceasta legatura functioneaza pe principiul UDP, astfel incat nici clientul, nici serverul, nu sunt interesati daca la celalalt capat a fost receptionat mesajul sau nu. In cazul in care cererea nu este primita de server sau clientul nu primeste raspunsul la cerere, incercarea de comunicare va fi in zadar.

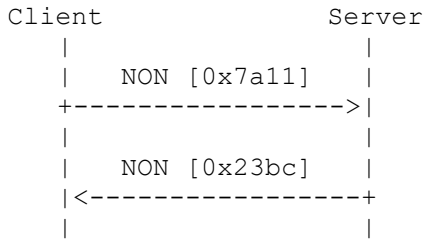

Figura 4 – Transmiterea mesajelor in cadrul CoAP (NON)

1.4 Descrierea proiectului (accesarea unui sistem de fisiere remote)

Aplicarea protocolului CoAP in cadrul proiectului (Browser FS)

In cadrul aplicatiei dezvoltate, folosirea protocolului CoAP consta in atasarea header-ului specific la mesajul pe care dorim sa-l trimitem si a carui structura (format) trebuie sa fie aceeasi (acelasi) atat pentru client, cat si pentru server. Pentru structura mesajului s-a stabilit un JSON cu 5 campuri pentru crearea cererii si un JSON cu 3 campuri la crearea raspunsului la cerere.

In cadrul aplicatiei:

- clientul va fi responsabil de -> crearea cererii
  -> parsarea raspunsului
  -> crearea si utilizarea interfetei grafice
- serverul va fi responsabil de -> parsarea cererii
  -> crearea raspunsului
  -> gestionarea sistemului de fisiere

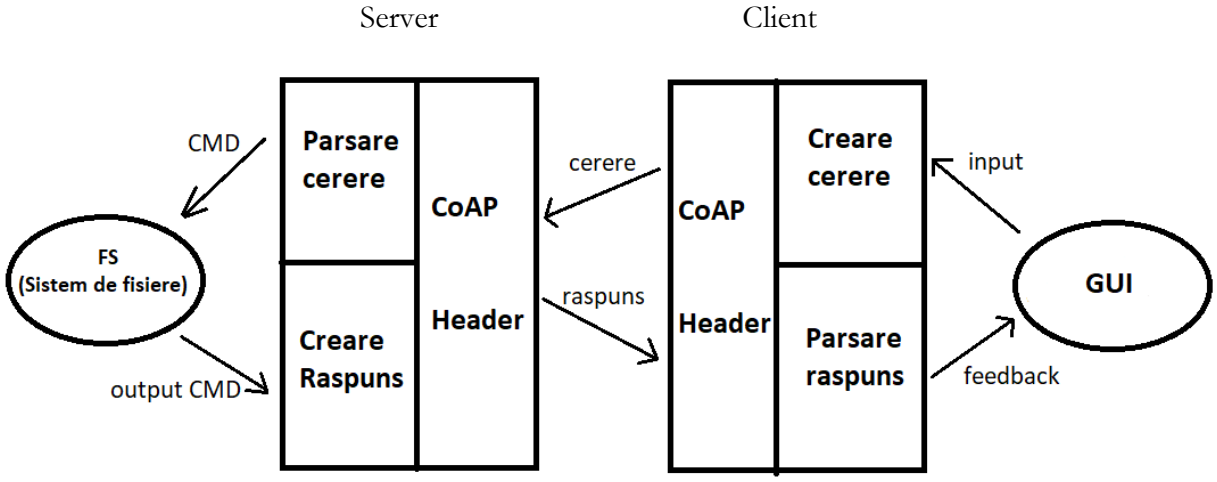

Fig. 4 – Schema de ansamblu a aplicatiei

Scopul proiectului

Proiectul are ca obiectiv crearea unei interfete grafice prin intermediul careia un client (utilizator) poate dialoga cu un server remote, efectuand diferite operatii asupra sistemului de fisiere gestionat de acesta din urma. Ca operatii posibile propunem crearea, stergerea, rularea (executarea) fisierelor, etc.

Comenzile propriu-zise pe care ne-am propus sa le implementam sunt:

- cd -> schimbarea directorului curent
- ls -> listarea fisierelor din directorul current
- pwd -> afisarea caii curente (de la Root pana la folderul current)
- msg -> transmiterea unui mesaj text serverului
- readText -> citirea continutului unui fisier
- run -> rularea unui fisier executabil (metoda speciala)
- newFile -> crearea unui nou fisier
- newDir -> crearea unui nou director
- rm -> sterge un fisier sau un director
- copy -> copierea unui fisier
- paste -> lipirea fisierului salvat de comanda copy la calea actuala
- write -> rescrierea unui fisier text
- append -> scrierea in continuarea unui fisier text

Ne propunem, de asemenea, sa utilizam urmatoarele coduri (specifice CoAP):

- 0.00 -> EMPTY (mesaj fara continut)
- 0.01 -> GET
- 0.02 -> POST
- 0.04 -> DELETE
- 0.21 -> RUN (metoda speciala -> rularea unui fisier executabil)
- 0.22 -> REGISTER
- 0.23 -> MESSAGE
- 2.01 -> Created
- 2.02 -> Deleted
- 2.05 -> Content
- 2.21 -> ConfirmExec
- 4.00 -> Bad Request
- 5.01 -> Not implemented

Dezvoltarea proiectului

Proiectul va fi dezvoltat folosind limbajul Python3 care ne va pune la dispozitie modulele “sockets”, pentru transmiterea pachetelor UDP si “os”.

De asemenea, se va folosi PyQt5 pentru crearea interfetei grafice, care va oferi multiple optiuni clientului in ceea ce priveste tipul requestului, selectarea mecanismului de comunicatie cu/fara confirmare, alegerea continutului mesajului si vizualizarea raspunsurilor primite din partea serverului.

1.5 CoAP vs HTTP

Un fapt cunoscut este acela ca atat HTTP, cat si CoAP sunt protocoale de transfer document. Spre deosebire de protocolul HTTP, pachetele trimise de catre CoAP sunt mult mai mici ca dimensiuni, acest lucru fiind benefic in utilizarea eficienta a memoriei RAM.

CoAP a fost introdus ca o alternativa mai simpla a protocolului HTTP pentru a conecta dispozitivele electronice limitate in specificatii la reteaua web. Unul dintre principalale avantaje ale utilizarii acestui protocol este utilizarea redusa a bateriei dispozitivelor, deoarece este necesar mai putin echipament hardware pentru comunicatie.

O alta deosebire dintre cele doua protocoale este ca HTTP utilizeaza modul de transfer TCP (Transmission Control Protocol), pe cand CoAP utilizeaza UDP (User Datagram Protocol). Acest lucru are un efect negativ asupra CoAP deoarece UDP necesita trimiterea unui ping la fiecare cateva secunde pentru a mentine conexiunea NAT/Firewall deschisa. In contrast, TCP obtine un timp mai indelungat de conexiune deschisa (cca. 15 minute) dupa apelarea instrumentului de retea ping (Packet Internet or Inter-Network Groper).

Ping-ul verifica daca un anumit calculator poate fi accesat prin intermediul unei retele de tip IP. Ping trimite mesaje de tip "echo request" (solicitare de raspuns) prin pachete adresate host-ului vizat si asteapta raspunsul la aceste mesaje venite sub forma de raspunsuri "echo response" de la hostul destinatie. Transmitand periodic astfel de pachete si calculand intarzierea cu care ajung raspunsurile, ping estimeaza timpul de raspuns, precum si rata de pierdere a pachetelor dintre host-uri.

Pentru o mai buna evidentiere a diferentelor dintre CoAP si HTTP, se observa urmatorul tabel:

Caracteristica	CoAP	HTTP
Protocol	Foloseste UDP	Foloseste TCP
Layer	Foloseste IPv6 cu 6LoWPAN	Foloseste IP
Suport multicast	Da	Nu
Comunicare sincrona	Nu este necesar	Este necesar

1.6 CoAP vs MQTT

MQTT, la fel ca CoAP, reprezinta un protocol de IoT (Internet of Things), astfel functionand la nivelul dispozitivelor cu un hardware mai redus. CoAP este un protocol de tip one-to-one, care permite comunicare intre 2 dispozitive, in timp ce MQTT permite comunicarea de tip many-to-many, adica intre mai multi utilizatori.

1.7 Interfata grafica:

Aplicatia de fata a fost implementata in limbajul de programare python si framework-ul PyQt5 pentru a realiza interfata grafica. Cu ajutorul GUI (Graphical User Interface), utilizatorul se poate conecta la un server CoAP ce asigura furnizarea informatiilor cerute de-a lungul sesiunii. Interfata este impartita sugestiv in 3 blocuri principale, numerotate corespunzator ordinii de acces. Pentru a avea acces la blocurile numerotate superior, va trebui sa fie validate blocurile anterioare.

Primul bloc impune conectarea clientului la server prin intermediul unui IP (IP-ul comunicat de server), un port sursa (prin intermediul caruia sa avem acces la datele primite) si un port destinatie (stabilit de server si comunicat ulterior clientilor). Pentru a utiliza aplicatia local, va trebui sa fie rulat serverul pe acelasi calculator, iar ip-ul va fi IPv4 asociat acelui calculator de catre routerul din retea. Pentru porturi, se vor considera valori in intervalul 1025 – 65535, deoarece primele 1024 porturi sunt private, asociate altor procese de catre sistemul de operare. In cazul in care formatul IP-ului sau intervalul de porturi sunt invalide, interfata va anunta utilizatorul de greseala comisa printr-un mesaj corespunzator in linia de comanda simulata in partea de jos a aplicatiei.

Al doilea bloc simuleaza etapa de conectare cu un cont pe server, prin introducerea unui nume de utilizator si a unei parole. Prin apasarea butonului “Set authentication” se vor memora valorile introduse in doua variabile ale programului, iar in momentul trimiterii unui pachet, se vor trimite si informatiile referitoare la cont, pentru a fi verificate de catre server. In cazul in care utilizatorul vrea sa se inregistreze, va fi nevoit, in aceeasi maniera, sa introduca datele despre noul sau cont la tastatura.

Cel de-al treilea bloc consta in compunerea unei cereri coerente pentru a utiliza sistemul de fisiere din spatele serverului. In functie de comanda dorita, pot aparea parametri atat in continuarea liniei “Command”, cat si in blocul de mai jos. Mai jos putem selecta tipul mesajului, care poate fi de tip Confirmable sau Nonconfirmable. In prima situatie, inainte de a primi mesajul propriu-zis de la server, vom primi un mesaj “gol” in care ne comunica ca au primit cererea noastra (mesaj de tip confirmabil). De asemenea, ei ne vor solicita in aceasta situatie sa confirmam ca am primit mesajul propriu-zis, prin trimiterea unui mesaj “gol” cu confirmare. In cel de-al doilea caz, dupa trimiterea cererii, serverul ne va trimite un singur raspuns ce va contine intregul mesaj cerut (nu se confirma primirea mesajului, deci, in cazul in care nu se primeste mesajul, va fi pierdut). Aceste informatii pot fi vizualizate si in figura 3 de mai sus. Daca utilizatorul nu stie exact formatul comenzilor ce pot fi trimise catre server, poate apasa butonul “Display All Commands”, afisandu-se astfel toate comenzile posibile in linia de comanda de mai jos.

Aceasta interfata grafica dispune de doua console care asigura comunicarea cu utilizatorul. Linia de comanda denumita “Command Prompt” comunica mesajele din consola, mai putin utile utilizatorului, de exemplu primirea unui mesaj. Consola din dreapta va afisa mesajele primite de server (payload-ul), formatat astfel incat sa fie usor de citit.

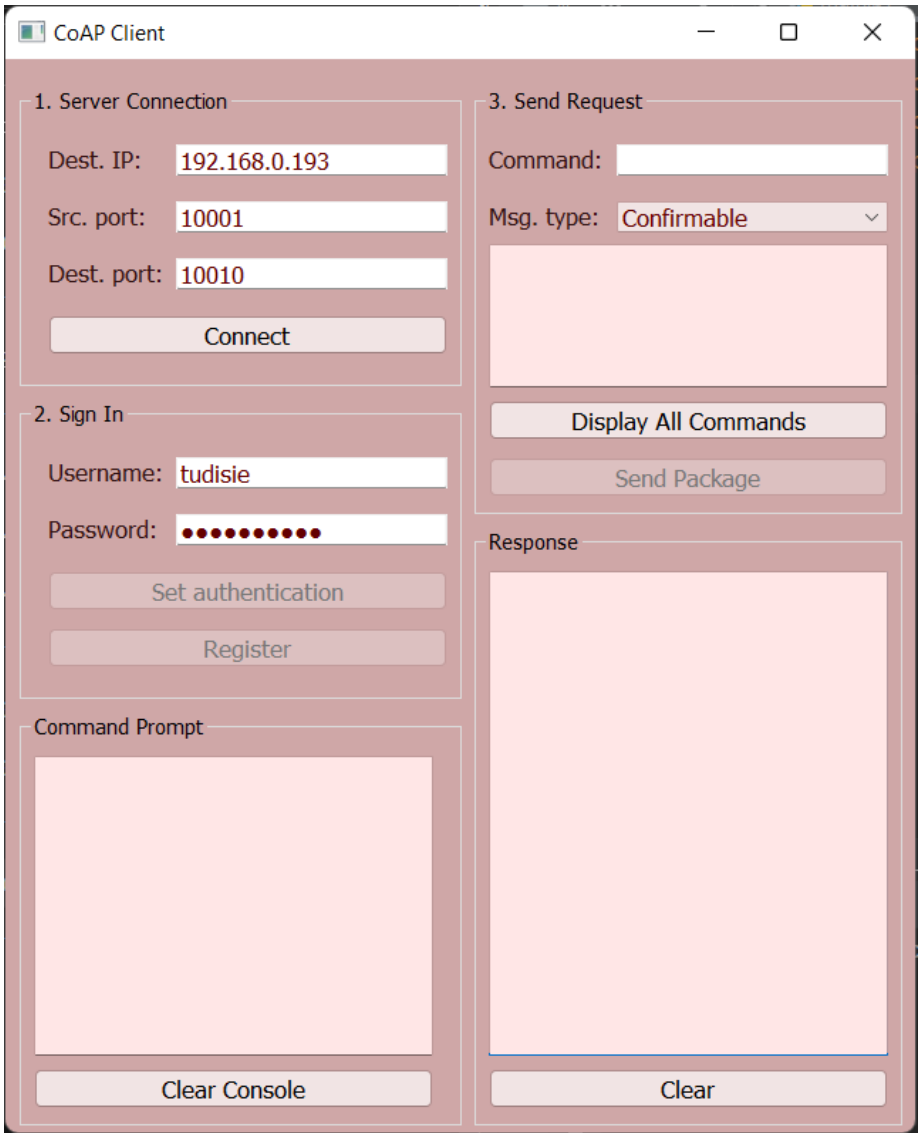

Fig. 5 – Interfata grafica a Clientului

1.8 Exemple:

Firefox are suport pentru protocolul CoAP folosind un plug-in numit Copper(CU).
Senzori. Acestia au o capacitate mica de memorare si o putere redusa de procesare.

1.9. Alte detalii de implementare:

Programul adauga o metoda speciala numita REGISTER si va fi asociata method code-ului 0.22. In momentul in care serverul primeste un pachet cu acest cod de raspuns, el va sti ca primeste o cerere de inregistrare a unui nou user. Serverul retine conturile intr-un fisier text cu formatul "username=parola". Daca contul este deja inregistrat in fisier, serverul va trimite un raspuns corespunzator clientului. Altfel, contul va fi introdus pe o linie noua daca respecta cerintele impuse de server. In momentul in care un utilizator trimite un mesaj de request, serverul verifica in baza lor de date daca user-ul respectiv exista sau nu.

Mesajele vor fi trimise cu ajutorul modulului socket, ce detine functia sendto() catre celalalt capat al liniei. Formatul ales este "latin-1", pentru ca acesta este interpretat corect de catre wireshark (spre deosebire de "UTF-8" si alte formate similare). Mesajul este impartit pe octeti, iar la primire, functia .receive() retine intr-un string de octeti toata informatia. Informatia astfel obtinuta trebuie prelucrata pe bucati: header-ul, respective payload-ul.

Payload-ul a fost ales de comun acord de catre server si client:

JSON-ul trimis de catre client are o structura formata din 5 campuri, de tipul:

```
{
        "username" : "Tudisie" ,
        "password" : "matiz4life" ,
        "command" : "ls" ,
        "parameters" : "-l" ,
        "timestamp" : "05/12/2021 – 13:11"
}
```

JSON-ul trimis de catre server are o structura formata din 3 campuri:

```
{
        "content" : "fisier.txt folder …" ,
        "error : "Eroare: Cerere refuzata ,
        "command" : "ls"
}
```

1.10 Bibliografie:

https://datatracker.ietf.org/doc/html/rfc7252#section-2.1
https://en.wikipedia.org/wiki/Constrained_Application_Protocol
https://www.ubicomp.org/ubicomp2013/adjunct/adjunct/p1495.pdf
https://www.rfwireless-world.com/Terminology/Difference-between-CoAP-and-HTTP.html
https://www.sciencedirect.com/topics/engineering/constrained-application-protocol