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".

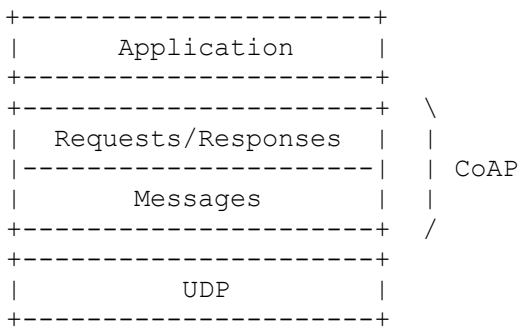


Fig. 1 - Structura CoAP

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 8 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
- 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 (poate avea intre 0 si 8 octeti)
- Request/Response Code (1 octet) -> message/response codes
- 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.
- Options -> reprezinta un camp pentru optiuni pe 8 biti, putand fi interpretat in diverse moduri, in functie de valorile celorlalte campuri.

Codul unei cereri/raspuns (8 biti):

0	1	2	3	4	5	6	7
Class			Code				

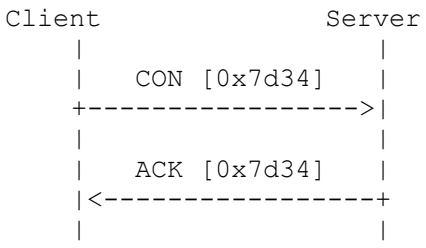
Cei mai semnificativi 3 biti formeaza un numar intitulat "Class", concept asemanator codurilor de status ale protocolului HTTP. Ultimii 5 biti din reprezentarea cererii sau a raspunsului reprezinta un cod care comunica mai multe detalii despre tipul de mesaj. Practic, mesajul poate fi comunicat sub forma "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:

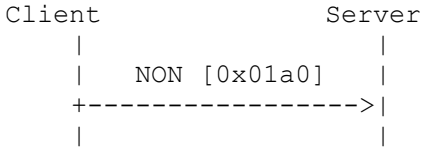
Class = 0.xx – request;
Class = 2.xx – success response;
Class = 4.xx – client error response;
Class = 5.xx – server error response;

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

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:





Figurile 3 – Transmiterea mesajelor in cadrul CoAP

1.4 Descrierea proiectului (accesarea unui sistem de fisiere remote)

Aplicarea protocolului CoAP in cadrul proiectului (Browser FS)

In cadrul aplicatiei dezvoltate de noi, 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. Ca exemplu, am putea scrie o cerere in format JSON, unde cheia este comanda, iar valoarea va enumera parametrii comenzii.

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

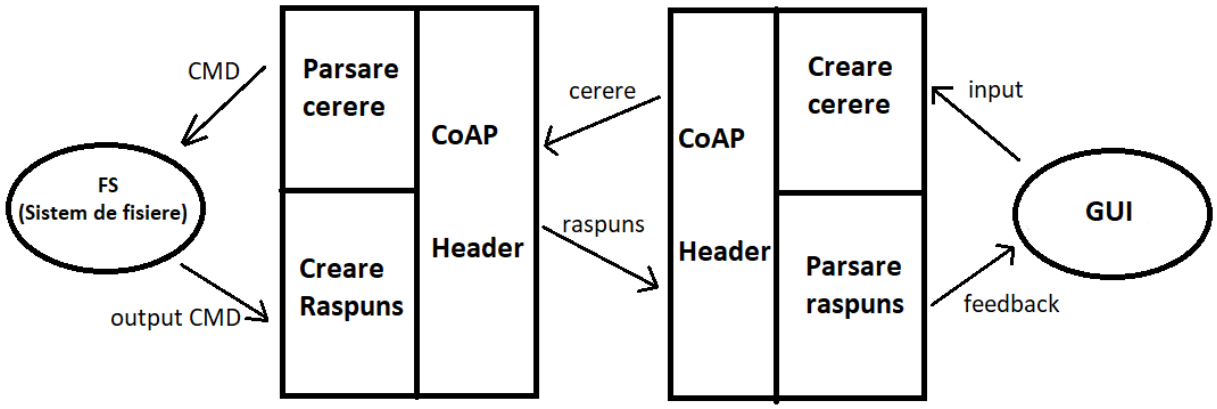


Fig. 4 – Schema de ansamblu a aplicatiei

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
- open -> deschidere continut fisier
- save -> salvare continut (eventual updatat) fisier
- newFile -> crearea unui nou fisier
- newDir -> crearea unui nou director
- ls -> listarea fisierelor din directorul curent

- rm -> sterge un fisier sau un director
- readText -> citirea continutului unui fisier text
- run -> rularea unui fisier executabil (metoda speciala)

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)
- 2.01 -> Created
- 2.02 -> Deleted
- 2.05 -> Content
- 2.21 -> ConfirmExec
- 4.00 -> Bad Request
- 5.01 -> Not implemented

De asemenea, vom 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.

Dezvoltarea proiectului

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

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 are 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 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.

1.8 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