



Aplicație de tip browser FS (Server CoAP)

Gîza David-Noel

Grupa 1309B

Țiganaș Ștefan-Gabriel

Grupa 1309B

1 Descrierea temei

Protocolul de aplicație constrînsă (CoAP) este un protocol specializat de aplicații Internet pentru dispozitive constrînse. Acest protocol permite acelor dispozitive constrînse numite „noduri” să comunice cu Internetul mai larg folosind protocoale similare. CoAP este conceput pentru a fi utilizat între dispozitive din aceeași rețea constrînsă (de exemplu, rețele cu consum redus de energie sau pierderi).

Ce este un nod de rețea? Un nod este orice dispozitiv fizic dintr-o rețea de alte dispozitive care poate trimite, primi și da mai departe informații. Computerul este cel mai des întâlnit nod și se numește adesea nodul computerului sau nodul de internet. Un nod de rețea este de obicei orice dispozitiv care primește și apoi comunică ceva prin rețea, dar poate primi și salva doar datele, transmitînd informații în altă parte sau poate crea și trimite date.

CoAP este un protocol de strat de serviciu care este destinat utilizării pe dispozitive de internet cu resurse limitate, cum ar fi nodurile de rețea ale senzorilor fără fir . CoAP este conceput pentru a se traduce cu ușurință în HTTP pentru o integrare simplificată cu web, îndeplinind în același timp cerințe specializate, cum ar fi suport multicast , cheltuieli generale reduse și simplitate.

Pachetele CoAP utilizează câmpuri de biți pentru a maximiza eficiența memoriei și folosesc în mod extensiv mapările din șiruri de caractere la numere întregi pentru a păstra pachetele de date suficient de mici pentru a transporta și a interpreta pe dispozitiv. Pe lângă dimensiunea pachetelor extrem de redusă, un alt avantaj major al CoAP este utilizarea UDP; folosirea de dateagrame permite rularea CoAP pe tehnologiile bazate pe pachete precum SMS.

User Datagram Protocol(UDP) este un protocol de comunicație pentru calculatoare care împreună cu Internet Protocol (IP), face posibilă livrarea mesajelor într-o rețea. Este similar cu sistemul poștal, în sensul că pachetele de informații (corespondența) sunt trimise în general fără confirmare de primire, în speranța că ele vor ajunge, fără a exista o legătură efectivă între expeditor și destinatar

Unele dintre cazurile specifice în care CoAP sunt utile sunt: Hardware-ul dvs. nu poate rula HTTP sau TLS: dacă acesta este cazul, atunci rularea CoAP și DTLS poate face practic același lucru ca HTTP. Dacă cineva este expert în API-uri HTTP, atunci migrarea va fi simplă. Primești GET pentru citire și POST, PUT și DELETE pentru mutații și securitatea rulează pe DTLS. Hardware-ul dvs. folosește bateria: dacă aceasta este o problemă, atunci rularea CoAP va îmbunătăți performanța bateriei în comparație cu HTTP prin TCP / IP. UDP economisește o anumită lățime de bandă și face protocolul mai eficient.

2 Detalii de implementare

- a)Implementarea unei soluții de accesare a unui sistem de fișiere (FS) la distanță(intre server si client).
- b)Server-ul trebuie sa puna la dispoziție resursele FS
- c)Cele două aplicații(server CoAP si client CoAp) trebuie să suporte un număr de coduri care să includă - codul 0.00 (mesaj fără conținut), metodele GET, POST (Method Codes) și o metodă nouă propusă de echipe în contextul temei, codurile de răspuns relevante pentru aplicație.
- d)Aplicațiile trebuie să suporte mecanismul de comunicație cu confirmare, cât și mesaje fără confirmare (selectabil din GUI).

Mesajul CoAP este cel mai mic strat și se ocupă cu schimbul de mesaje UDP între puncte finale. Mesajul CoAP are un ID unic și trei părți:

1. un antet binar (32 byte)
2. o serie de opțiuni compacte
3. sarcină utilă

Protocolul CoAP utilizează două tipuri de mesaje: mesaj confirmabil, mesaj neconfirmabil.

Un mesaj confirmabil CoAP este un mesaj de încredere. În timpul schimbului de mesaje între două puncte finale, aceste mesaje pot fi fiabile. În protocolul CoAP, un mesaj de încredere este obținut folosind un mesaj confirmabil (CON). Folosind acest tip de mesaj,

clientul poate fi sigur că mesajul va ajunge la server. Un mesaj de confirmare CoAP este trimis din nou și din nou până când cealaltă parte trimite un mesaj de confirmare (ACK). Mesajul ACK conține același ID al mesajului confirmabil (CON).

Mesaje neconfirmabile (NON) care nu necesită o confirmare de către server. Aceste mesaje sunt mesaje nesigure, nu conțin informații critice care trebuie livrate serverului. La această categorie aparțin mesaje care conțin valori citite de senzori. Chiar dacă aceste mesaje nu sunt fiabile, au un ID unic. Acesta este al doilea strat din stratul de abstractizare. Aici cererea este trimisă utilizând un mesaj Confirmable (CON) sau Non-Confirmable (NON). Există mai multe scenarii în funcție de posibilitatea serverului de a răspunde imediat la solicitarea clientului sau răspunsul dacă nu este disponibil: Dacă serverul poate răspunde imediat la solicitarea clientului, atunci dacă cererea este efectuată utilizând un mesaj confirmabil (CON), serverul trimite înapoi clientului un mesaj de confirmare care conține răspunsul sau codul de eroare.

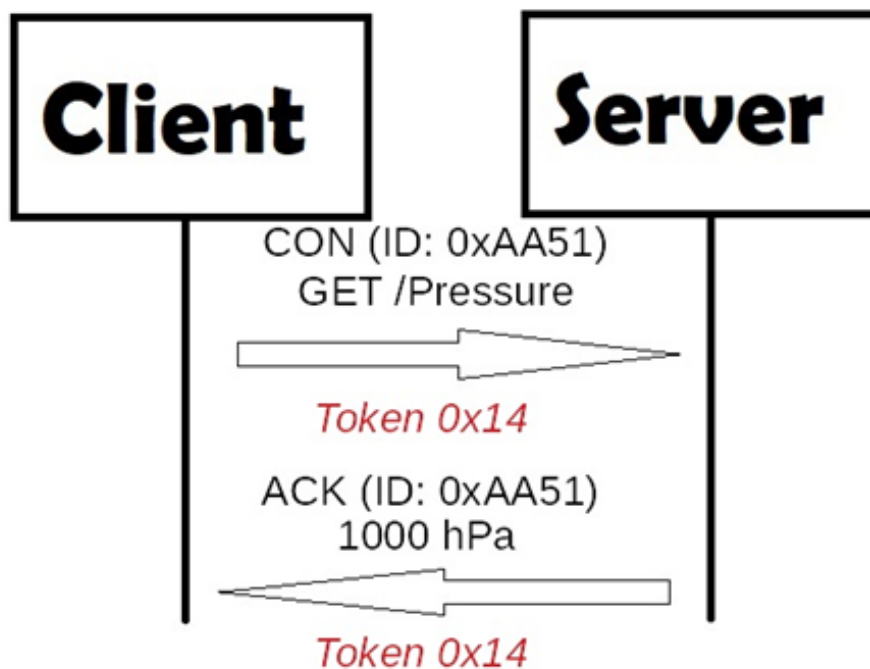


Fig.01 Transmitere mesaj CoAP(cu confirmare)

Aici Tokenul este diferit de ID-ul mesajului și este utilizat pentru a se potrivi cu cererea și răspunsul. Dacă serverul nu poate răspunde la cerere, atunci serverul trimite o confirmare cu un răspuns gol. De îndată ce răspunsul este disponibil, serverul trimite un nou mesaj confirmabil către client care conține răspunsul. În acest moment, clientul trimite înapoi un mesaj de confirmare.

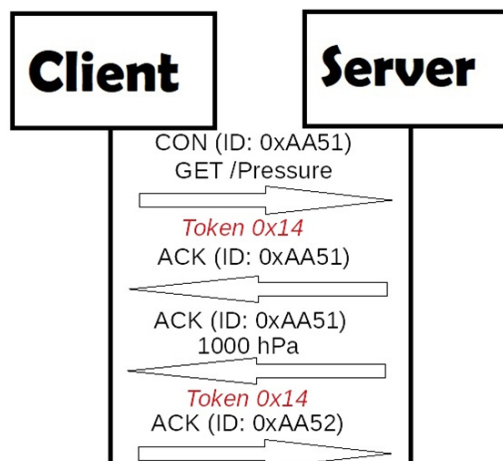


Fig.02 Transmitere mesaj CoAP(cu confirmare si ACK)

Formatul mesajului:

Ver	T	TKL	Code	Message ID
Token				
Options (if exists..)				
Payload (if exists..)				

Fig.03 Formatul unui mesaj CoAP

Unde:

1. Ver: Este un număr întreg desemnat de 2 biți care indică versiunea
2. T: este un număr întreg de 2 biți desemnat care indică tipul mesajului: 0 confirmabil, 1 neconfirmabil;
3. TKL: Lungimea jetonului este lungimea jetonului de 4 biți
4. Cod: este răspunsul la cod (lungimea de 8 biți)
5. ID mesaj: este ID-ul mesajului exprimat cu 16 biți;

3 Prezentarea metodelor(Get, Put, Post, Delete)

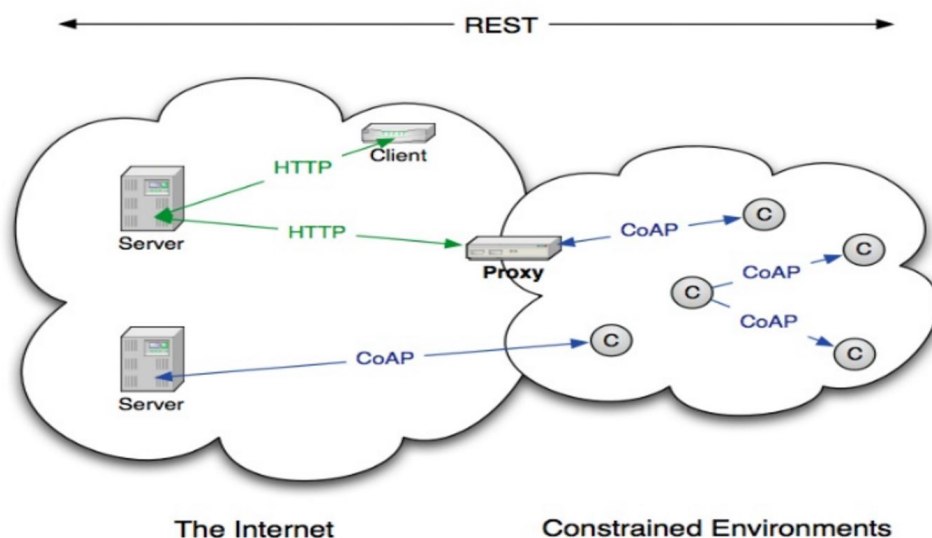
1. Metoda GET preia o reprezentare pentru informațiile care în prezent corespund resursei identificate prin URI-ul de solicitare. Dacă solicitarea include o opțiune de acceptare, aceasta indică formatul de conținut preferat al unui răspuns. Dacă cererea include Opțiunea ETag, metoda GET solicită validarea ETag și reprezentarea să fie transferată numai dacă validarea a eșuat. Pentru succes în răspuns va fi un cod de răspuns 2.05 (conținut) sau 2.03 (valid). Metoda GET este sigură și idempotentă.

2. Metoda PUT solicită ca resursa identificată prin cerere URI-ul să fie actualizat sau creat cu reprezentarea anexată. Formatul de reprezentare este specificat de tipul de conținut și conținut, codarea dată în opțiunea format-conținut, dacă este furnizată. Dacă există o resursă la cererea URI, reprezentarea anexată trebuie să fie considerată o versiune modificată a resursei respective și o versiune 2.04 (Modificat) codul de răspuns trebuie returnat. Dacă nu există resursă serverul poate crea o nouă resursă cu acel URI, rezultând un cod de răspuns 2.01 (creat). Dacă resursa nu a putut fi creată sau modificată ar trebui să fie trimis un cod de răspuns de eroare adecvat. PUT nu este sigur, dar este idempotent.
3. Metoda POST solicită ca reprezentarea inclusă în cerere să fie procesată. Funcția efectivă a metodei POST este determinată de serverul de origine și depinde de resursă țintă. De obicei, rezultă crearea unei noi resurse sau actualizarea resursei țintă. Dacă a fost creată o resursă pe server, răspunsul returnat de către server trebuie să aibă un cod de răspuns 2.01 (creat). Dacă POST reușește dar nu are ca rezultat crearea unei noi resurse serverul trimite un cod de răspuns 2.04 (modificat). POST nu este nici sigur, nici idempotent.
4. Metoda DELETE realizează stergerea unei resurse. Un cod de răspuns 2.02 (șters) reprezintă succesul operației sau în cazul în care resursa nu a existat înainte de cerere. DELETE nu este sigur, dar este idempotent.

O metodă HTTP este idempotentă dacă se poate face o cerere identică o dată sau de mai multe ori la rând cu același efect, lăsând serverul în aceeași stare. Cu alte cuvinte, o metodă idempotentă nu ar trebui să aibă efecte secundare (cu excepția păstrării statisticilor).

Aplicațiile trebuie să suporte mecanismul de comunicație cu confirmare, cât și mesaje fără confirmare (selectabil din GUI) -pe GUI vom crea un checkbox pentru mecanismul de confirmare/neconfirmare a mesajului; -Pentru conectarea UDP-ului la stiva de comunicații vom folosi biblioteca din Python: Socket, iar pentru realizarea interfeței grafice vom folosi Tkinter.

The CoAP Architecture



16

Fig. 04 HTTP-CoAP

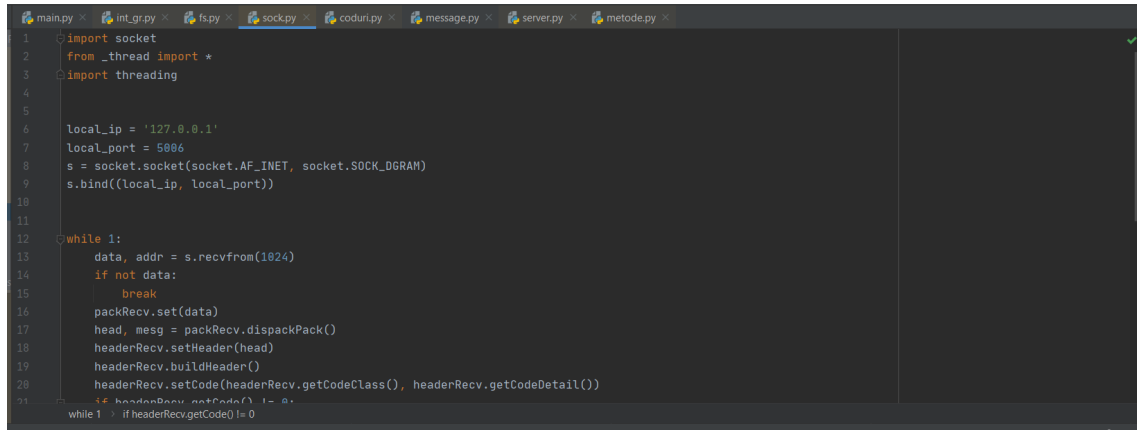
4 Modalitatea de lucru propusă

Identificarea și alocarea task-urilor

Task ID	Descriere task	Membru echipă
task1	Documentatie tema	m1, m2
task2	Descriere protocol CoAp	m1
task3	Schelet aplicatie	m2
task4	Implementarea interfetei	m1, m2
task5	Implementare metode(method codes)+cod	m1, m2
task6	Testare aplicatie	m1, m2
task7	Raport final	m1, m2

Git repository: <https://github.com/davidgiza5/RC-P>

5 Conectare server

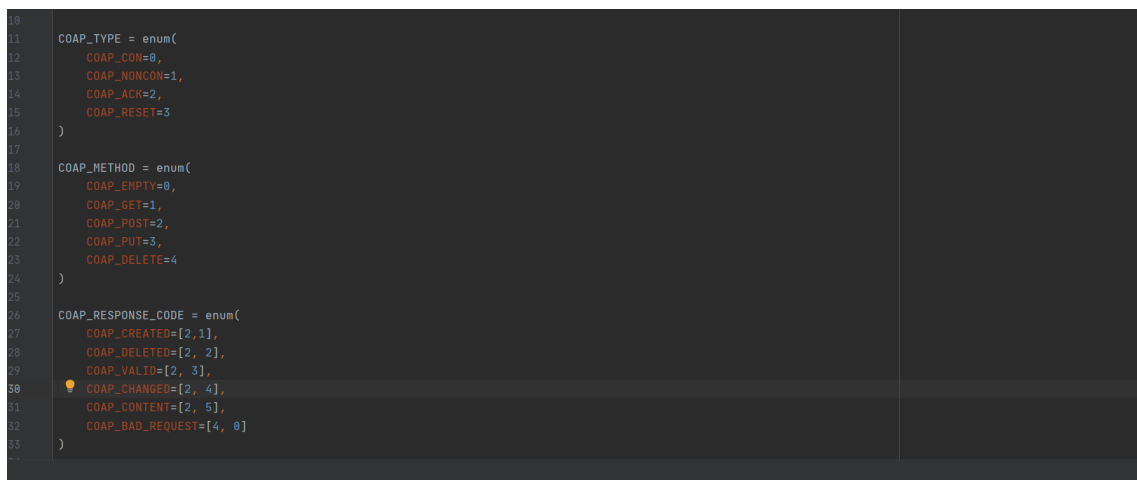


```
1 import socket
2 from _thread import *
3 import threading
4
5
6 local_ip = '127.0.0.1'
7 local_port = 5684
8 s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
9 s.bind((local_ip, local_port))
10
11
12 while 1:
13     data, addr = s.recvfrom(1024)
14     if not data:
15         break
16     packRecv.set(data)
17     head, msg = packRecv.unpack()
18     headerRecv.setHeader(head)
19     headerRecv.buildHeader()
20     headerRecv.setCode(headerRecv.getCodeClass(), headerRecv.getCodeDetail())
21     if headerRecv.getCode() != 0:
22         while 1: if headerRecv.getCode() != 0:
```

Fig. 04 Conectare la Server CoAP

Pentru conectarea la server s-a folosit libraria socket din Python care este utilizata pentru a trimite mesaje intr-o retea. S-au folosit functii specifice acestei librarii, precum "socket()" (pentru crearea unui socket), "bind", pentru asocierea unei adrese specifice protocolului de retea utilizat, "connect", "recv" etc.

6 Formatul mesajului CoAP



```
10
11 COAP_TYPE = enum(
12     COAP_CON=0,
13     COAP_NONCON=1,
14     COAP_ACK=2,
15     COAP_RESET=3
16 )
17
18 COAP_METHOD = enum(
19     COAP_EMPTY=0,
20     COAP_GET=1,
21     COAP_POST=2,
22     COAP_PUT=3,
23     COAP_DELETE=4
24 )
25
26 COAP_RESPONSE_CODE = enum(
27     COAP_CREATED=[2, 1],
28     COAP_DELETED=[2, 2],
29     COAP_VALID=[2, 3],
30     COAP_CHANGED=[2, 4],
31     COAP_CONTENT=[2, 5],
32     COAP_BAD_REQUEST=[4, 0]
33 )
```

Fig. 05 Format header mesaj

Mesajele specifice CoAP sunt formate din versiune (numar intreg pe 2 biti), tip (mesaj-confirmabil/neconfirmabil), token (4 biti, utilizat pentru asocierea unei cereri cu un raspuns), code (8 biti pentru raspuns, 5 biti clasa, 3 biti detalii) MessageId (16 biti pentru detectia mesajelor

duplicate, respectiv asocierea perechilor de mesaje CON-ACK, CON-RST, NON-RST)

7 Definire header

```
1 from random import randint
2
3
4 class Mesaj:
5     def __init__(self):
6         self.header = ""
7         self.mesaj = ""
8         self.token = ""
9         self.pachet = ""
10
11     def set(self, pachet):
12         self.pachet = pachet
13
14     def createPachet(self, header, mesaj):
15         self.header = header
16         self.mesaj = mesaj
17         self.token = randint(0, 65536)
18         self.pachet = (" " + str(header.header)).encode('UTF-8')
19         if 0 < header.getTokenLength() <= 8:
20             self.pachet = self.pachet+(str(self.token)).encode('UTF-8')
21         if mesaj != "":
22             self.pachet = self.pachet+(str(self.mesaj)).encode('UTF-8')
23         return self.pachet
24
25     def getP(self):
26         return self.pachet
```

Fig. 06 Definire Header

8 Concluzii

In concluzie, CoAP (Constrained Application Protocol) este un protocol folosit de nodurile de retea constranse(energie redusa, consum redus). Totodata, CoAP este o extensie(fiind si compatibil) a HTTP pentru dispozitive constranse, acesta fiind conceput pentru fiabilitate in latime de banda redusa si congestie ridicata.

Protocolul foloseste impachetare UDP(USER DATAGRAM PROTOCOL), acesta fiind un protocol fara conexiune, utilizat in video streaming, transimisiuni in direct etc. De asemenea, fata de TCP, UDP este mai rapid si are nevoie de mai putine resurse, fiind mai potrivit pentru aplicatii care necesita transmisiune rapida si eficienta, cum ar fi jocurile video.

9 Referințe

<https://ro.qaz.wiki/wiki/ConstrainedApplicationProtocol>

<https://tools.ietf.org/id/draft-ietf-core-coap-04.xmlget>

<https://tools.ietf.org/html/rfc7252>