# UNIVERSITATEA TEHNICĂ ,,GHEORGHE ASACHI” IAȘI

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI**

Aplicație de tip browser FS

Client CoAP

**Coordonator, Studenți,**

**Nicolae-Alexandru Mihalache Nicoleta-Ecaterina**

**Botezatu Grupa: 1309A**

**Petrov Sergiu**

**Grupa: 1309B**

**Iași, 2020**

**1.Descrierea temei**

**CoAP** (Constrained Application Protocol)- este un protocol specializat pentru dispozitive constrânse, având ca scop conectarea la Internet a dispozitivelor ce dispun de resurse limitate (de calcul, memorie, alimentare etc.). Acesta este o simplificare a protocolului HTTP, fiind compatibil cu protocolul UDP.

Principalele caracteristici ale protocolului CoAP sunt:

* Este un protocol web utilizat in M2M cu cerințe constrânse;
* Realizează un schimb de mesaje asincron;
* Este simplu de analizat;
* URI și suport de conținut;

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

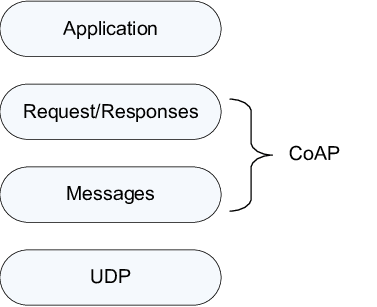


Fig.1. Stratificarea abstractă CoAP

Protocolul CoAP utilizează două tipuri de mesaje:

* Mesaj confirmabil;
* Mesaj neconfirmabil;

**Mesajul confirmabil (CON)** este un mesaj de încredere. Folosind acest tip de mesaj, cleintul 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) trimis de client.

**Mesajul neconfirmabil (NON)** este un mesaj ce nu necesită confirmare de la server. Este un mesaj nesigur ce nu conține informații critice care trebuie livrate la server.

**Formatul mesajului**:

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|Ver| T | TKL | Code | Message ID |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Token (if any, TKL bytes) ...

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Options (if any) ...

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|1 1 1 1 1 1 1 1| Payload (if any) ...

Fig.2.Formatul mesajului

Unde:

1. **Ver-** indică numărul versiunii CoAP și este reprezentată pe 2 biți.
2. **T-** indică tipul de mesaj al datagramei pentru contexul celor 2 tipuri de mesaje din cerere și răspuns (Cerere: 0=confirmabil și 1=non-confirmabil, Răspuns: 2=confirmare și 3=resetare) și este reprezentat pe 2 biți.
3. **TKL-** indică lungimea câmpului token cu lungime variabilă, care poate avea o lungime de 0-8 biți.
4. **Code-** indică codul de solicitare/ răspuns (8 biți): cei mai semnificativi 3 biți formează un număr cunoscut numit ,,clasă”, iar cei 5 biți mai puțin semnificativi formează un cod care comunică detalii suplimentare despre cerere sau răspuns.
5. **Message ID-** indică ID-ul mesajului reprezentaat pe 16 biți, folosit pentru a detecta duplicarea mesajelor și pentru a potrivi mesajele de tip Confirmare/ Resetare la mesajele de tip Confirmabil/ Non-confirmabil. Mesajul de răspuns va avea același ID de mesaj ca și cererea.
6. **Token-** indică un câmp opțional a cărui dimensiune este indicată de TKL și este folosit pentru a asocia o cerere cu un răspuns. Valorile sale sunt generate doar de client.
7. **Options-** indică un câmp opțional pentru opțiune, reprezentat pe 8 biți.
8. **Payload-** indică un câmp opțional pentru data utilă.

**Transmisia mesajelor Client- Server**:

Client Server

| |

| CON [0x7d34] |

+----------------->|

| |

| ACK [0x7d34] |

|<-----------------+

| |

Fig.3. Transmisie fiabilă a mesajelor

Fiabilitatea este asigurată prin marcarea unui mesaj ca Confirmabil (CON). Mesajul confirmabil este retransmis utilizând un timeout implicit și back-off exponențial între retransmisii până la destinatar care va trimite înapoi un mesaj de confimare (ACK) cu același Id de mesaj. Atunci când un destinatar nu este deloc capabil să proceseze un mesaj confirmabil acesta răspunde cu un mesaj de rest (RST) în loc de confirmare (ACK).

Client Server

| |

| CON [0xbc90] |

| GET /temperature |

| (Token 0x71) |

+----------------->|

| |

| ACK [0xbc90] |

| 2.05 Content |

| (Token 0x71) |

| "22.5 C" |

|<-----------------+

| |

Fig.4.Cerere GET cu răspuns

Atunci când serverul răspunde imediat la solicitarea clientului acesta trimite un mesaj confirmabil (CON) ce conține si data țintă. ID-ul mesajului este același, precum și valoarea Token-ului.

Client Server

| |

| CON [0x7a10] |

| GET /temperature |

| (Token 0x73) |

+----------------->|

| |

| ACK [0x7a10] |

|<-----------------+

| |

... Time Passes ...

| |

| CON [0x23bb] |

| 2.05 Content |

| (Token 0x73) |

| "22.5 C" |

|<-----------------+

| |

| ACK [0x23bb] |

+----------------->|

| |

Fig.5. Cerere GET cu un răspuns separat

Dacă serverul nu poate răspunde la cererea clientului, atunci el trimite o confirmare cu un răspuns gol. De îndată ce răspunsul este disponibil, server-ul trimite un nou mesaj confirmabil către client care conține răspunsul. În acest moment, client trimite înapoi un mesaj de confirmare.

**2.Detalii implementare**

Clientul trimite o cerere (confirmabilă/neconfirmabilă) server-ului în urma căreia acesta îi trimite datele/ pachetele . Dacă după un timp de la trimiterea cererii clientul nu primește niciun răspuns de la server atunci acesta va trimite din nou cererea până la un anumit interval de timp. Dacă serverul primește cererea de la client, atunci trimite un ACK către acesta, împreună cu datele sau trimite ACK-ul și după un interval de timp și datele.

Clientul va fi capabil să execute toată gama de operații standard (acces, modificare) pentru foldere și fișiere.

Operațiile standard pe care clientul le execută:

1)Acces-metoda GET

2)Modificare- metoda PUT

Aplicația suportă un număr de coduri care include și codul 0.00 (mesaj fără conținut) și metodele GET, POST.

Metoda GET- preia o reprezentare pentru informațiile care corespund în prezent resursei identificate de URI-ul de solicitare. Pentru succes codurile sunt: 2.05 (Conținut) sau 2.03 (Valid).

Metoda POST-solicită procesarea reprezentării incluse în cerere rezultând crearea unei noi resurse sau actualizarea acesteia. Dacă a fost creată o resursă pe server returnează codul 2.01 (Creat), dacă s-a reușit modificarea se returnează codul 2.04 (Modificat), iar dacă resursa țintă a fost ștearsă se returnează un cod de răspuns 2.02 (Șters).

Aplicația suportă mecanismul de comunicație cu confirmare și fără confirmare (selectabil din GUI).

Pentru conectarea UDP-ului la stiva de comunicații am folosit biblioteca din Python: Socket, iar pentru realizarea interfeței grafice am folosit Tkinter.

**3.Modalitatea de lucru propusa**

Identificarea și alocarea task-urilor

|  |  |  |
| --- | --- | --- |
| **Task ID** | **Descriere task** | **Membru echipă** |
| Task1 | Documentație temă | m1,m2 |
| Task2 | Descriere protocol CoAP | m1 |
| Task3 | Scheletul aplicației | m2 |
| Task4 | Implementarea interfeței | m1 |
| Task5 | Implementare metode+cod | m1, m2 |
| Task6 | Testare aplicație | m2 |
| Task7 | Raport final | m1, m2 |

Referințe:

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

<https://medium.com/@harshhvm/what-is-coap-protocol-coap-protocol-introduction-overview-3e8bac4d7f8e>

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