SERVER COAP

RC proiect

Membrii echipei:

Fechet Ionela Paula

Radu Cosmina Mihaela

Grupa 1309B

Ce este CoAP?

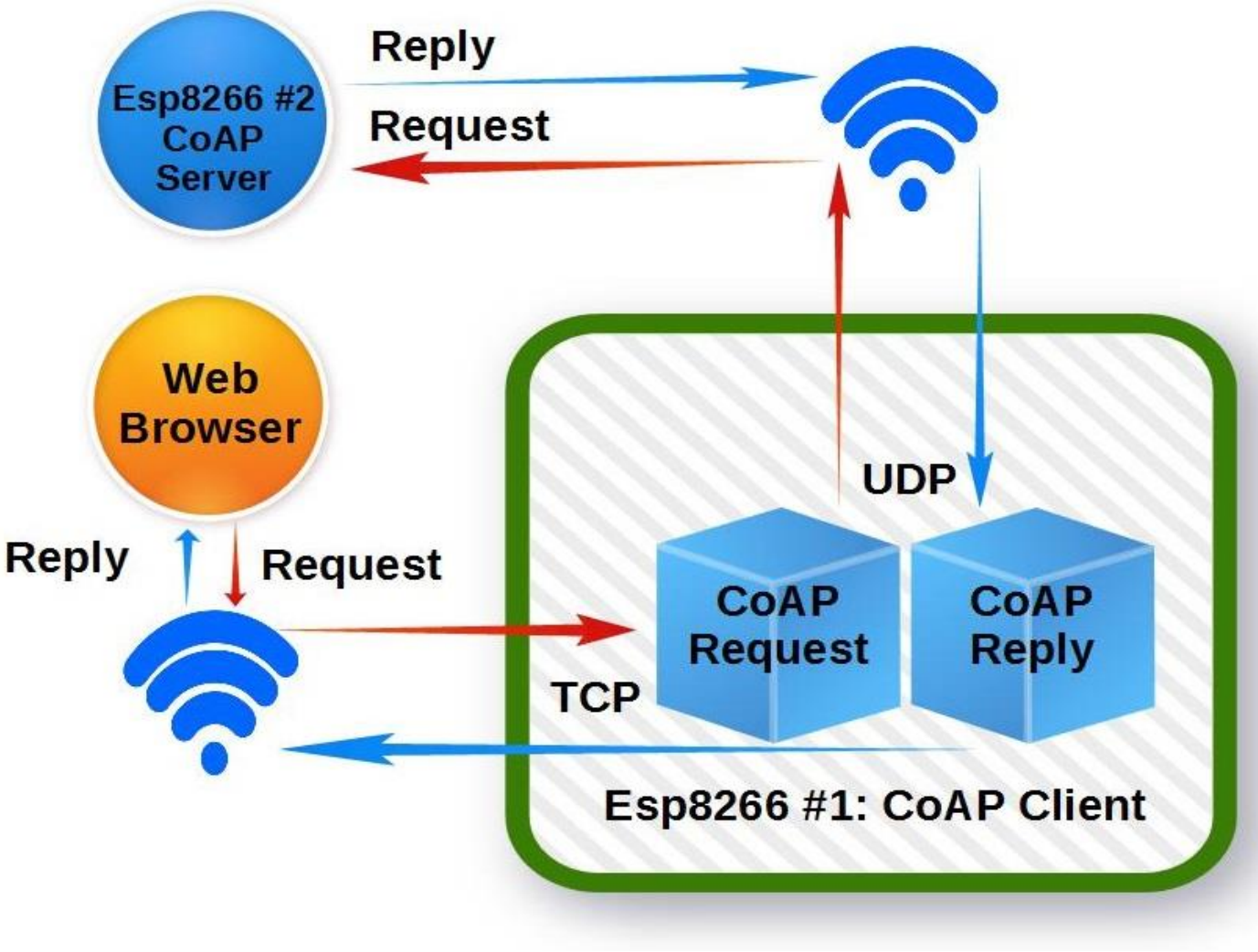
CoAP este un protocol IoT (Internet of things). CoAP reprezintă un protocol de aplicație restrâns și este definit în RFC 7252.

Este un protocol simplu, special conceput pentru dispozitive restrânse (cum ar fi microcontrolere) și rețele restrânse (low power). Acest protocol este utilizat în schimbul de date pentru aplicatii M2M (machine-to-machine) și este foarte similar cu **HTTP**.

CoAP oferă un model de interacțiune cerere / răspuns si include concepte cheie ale Web, cum ar fi URI-uri.

Ca și HTTP, CoAP se bazează pe modelul REST : serverele pun la dispoziție resursele sub o adresă URL, iar clienții accesează aceste resurse folosind metode precum GET, PUT, POST și DELETE.

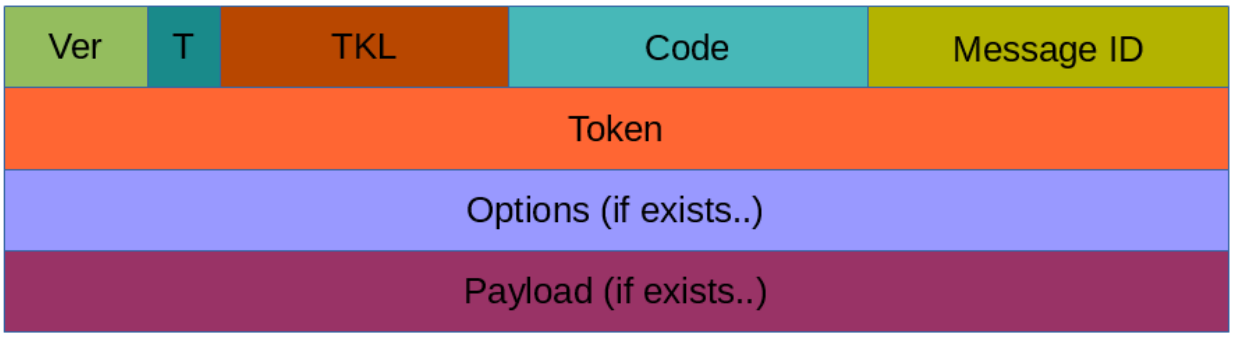
Mesajele CoAP sunt trimise folosind UDP, care prin natura nu este reliable, iar pentru a asigura securitate mesajelor se foloseste DTLS in schimb de UDP. Cererea este trimisă folosind un mesaj Confirmable (CON) sau Non Confirmable (NON). Există mai multe scenarii în funcție dacă serverul poate răspunde imediat la solicitarea clientului sau daca răspunsul nu este disponibil.



CoAP se bazează pe aspecte de securitate UDP (**User Datagram Protocol**) pentru a proteja informațiile. Deoarece HTTP folosește TLS (protocol care ofera confidențialitate și integritatea datelor între două sau mai multe aplicații) peste TCP (este o colecție

de algoritmi care extind UDP pentru a-l susține în ordinea livrării fluxurilor), CoAP folosește TLS peste UDP.

In proiectarea acestei aplicatii am urmarit construirea unui pachet CoAP urmarind structura de mai jos:



Cel mai mic mesaj CoAP are o lungime de 4 bytes, iar acesti 4 bytes sunt obligatorii in implementare.

COAP_MESSAGE_FORMAT

HEADERUL FORMAT DIN :

1. VERSIUNEA: 2 BITI

2. MESSAGE-TYPE : 2 BITI

3. TOKENLENGTH: 4 bit

4. CODE: -code class -3 biti

-code response -5 biti

CODE RESPONSE (8 bit)

5. MESSAGE_Id : 16 bit!

6. TOKEN : Dimensiunea tokenului este indicata de dimensiunea lui tokenlength, a carei valoare este generata de catre client. Serverul trebuie sa trimita inapoi clientului acelasi token.

7. Options

8. Payload : in cazul nostru, playload o sa fie orasul pentru care clientul doareste sa afle temperatura

EMPTY = <Code 0 “EMPTY”>

GET = <Request Code 1 “GET”>

POST = <Request Code 2 “POST”>

CONVERT = <Request Code 6 „CONVERT”>

CoAP GET:

Status codes pentru CoAP sunt : 2.03 pentru un request valid, 2.05 pentru un request valid cu content, iar 4.05 pentru cele care nu sunt permise. In cazul aplicatiei noastre, am returnat codul 2.05 cand totul a mers corect.

CoAP POST:

Am creat un fisier in care tinem datele despre orasele cu tot cu temperaturile lor. In cazul in care clientul alege sa faca post, data pentru orasul introdus va fi cautat in fisier. Daca orasul se gaseste deja in fisier, se modifica temperatura cu temperatura curenta (2.04), daca nu, se face un create (2.01).

Un alt cod returnat de catre aceasta metoda este 4.05, daca metoda nu este permisa.

Method Convert:

Temperatura returnata in urma requestului facut de server catre API este masurata in Kelvin. Urmarim sa convertim unitatea de masura a temperaturii in grade Celsius. Clasa este 0, iar codul este 6. Metoda Convert este 0.6.

Sunt 2 trasaturi specifice CoAP: **Messages** si **Request/Response** (ca in figura). Blocul „Messages" are de a face cu UDP si cu mesajele asincrone.

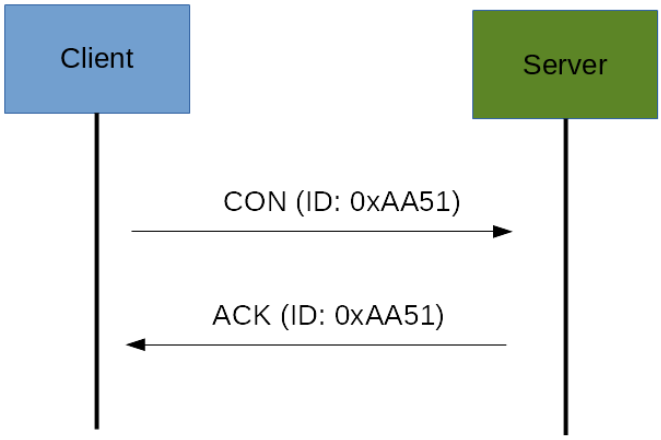
CoAP suporta patru tipuri diferite de mesaje:

1. -Confirmable
2. -Non-confirmable
3. -Acknowledgment
4. -Reset

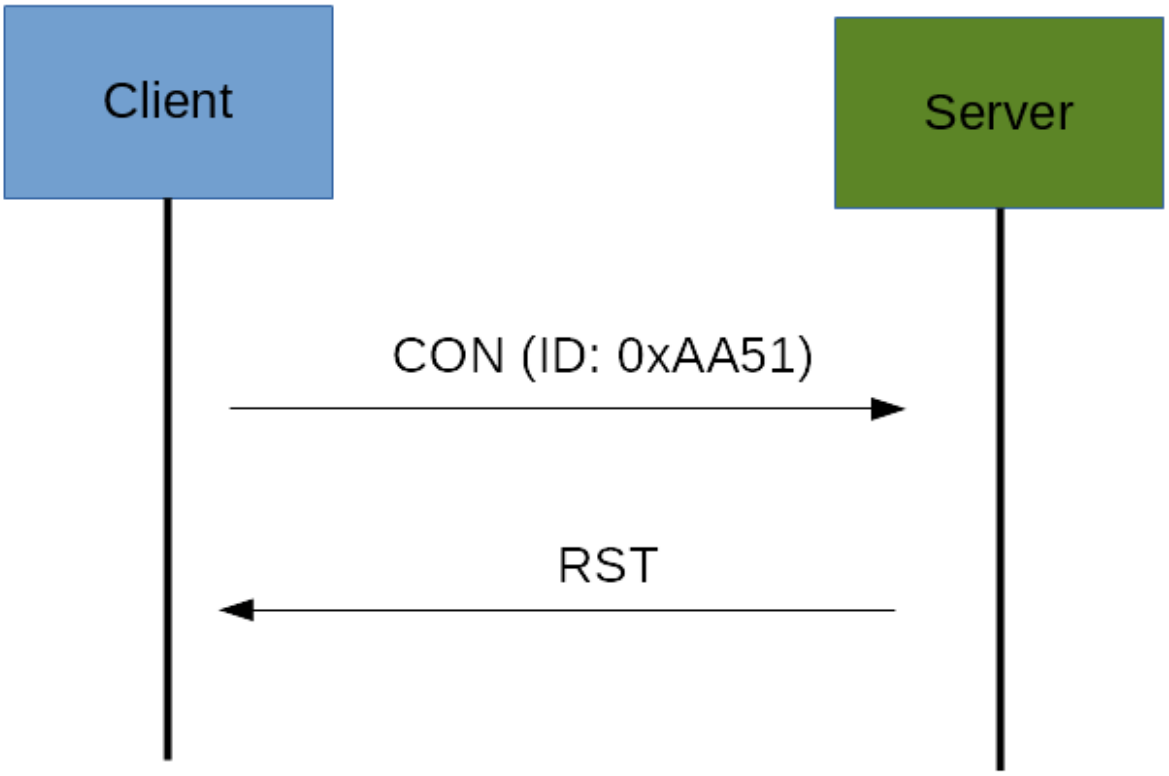
CoAP Messages Model

Fiecare mesaj CoAP are un id unic, este folositor pentru a detecta mesaje duplicate.

Un **mesaj confirmabil** este un mesaj reliable. Cu acest tip de mesaj confirmabil, clientul poate trimite din nou si din nou celeilalte parti pana cand primeste un acknowledgement answer (ACK). **Serverul va primi in mod sigur mesaj de la client.**

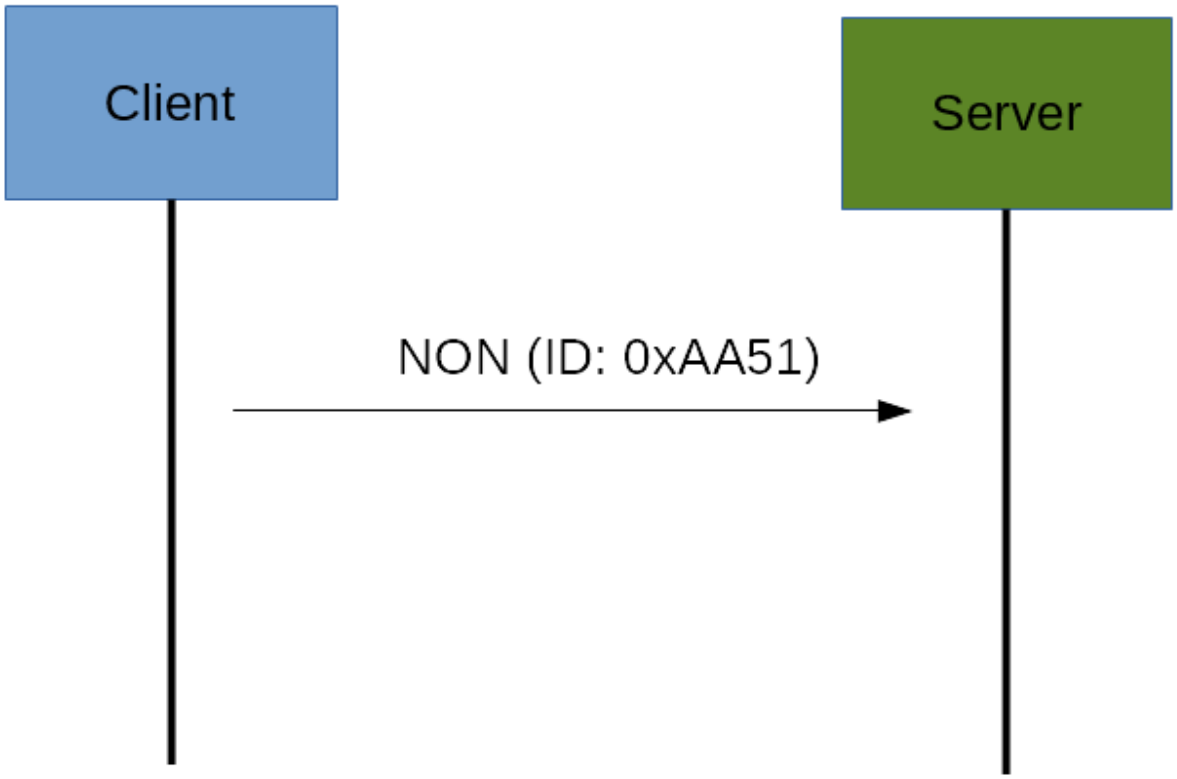


Daca serverul nu a putut sa managerieze mesajul primit de la client, acesta va trimite un mesaj reset (RST) in loc de ACK.



O alta categorie de mesaje este cel non-confirmabil. Mesajul trimis de catre client la server care nu contine informatii critice care ar trebui sa ajunga la server. Acest tip de mesaje nu necesita un acknowledge de la server.

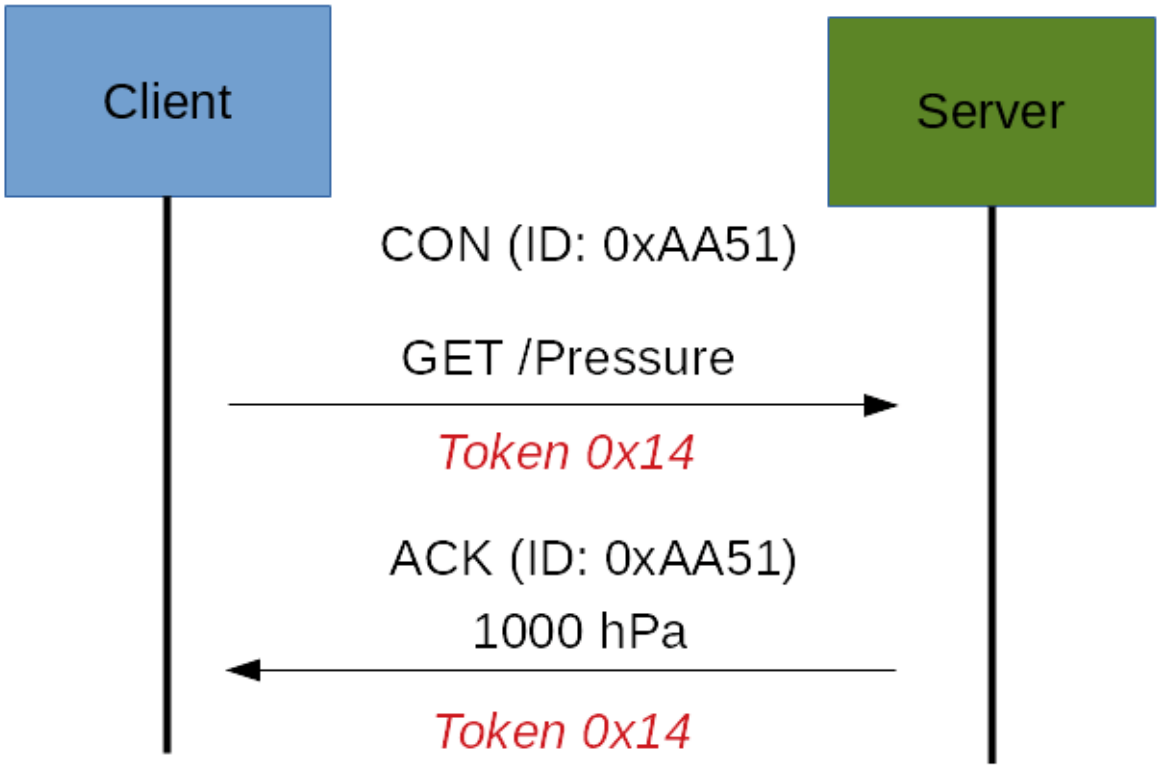
Dacă serverul poate răspunde imediat la solicitarea clientului, atunci dacă solicitarea este efectuată folosind un mesaj Confirmabil (CON), serverul trimite înapoi clientului un mesaj de Confirmare care conține răspunsul sau codul de eroare.



CoAp Request/Response Model

Requestul este trimis folosind mesaje confirmabile (CON) sau non-confirmabile (NON).

Daca mesajul este trimis la server cu un mesaj confirmabil, si serverul poate raspunde imediat la request, serverul va trimite inapoi un ACK, cu un raspuns sau cod de eroare.



Tokenul nu este aceeasi cu Message ID-ul si este folosit pentru a face legatura dintre request si raspuns.

Daca serverul nu poate raspunde imediat, el va trimite un ACK catre client cu mesaj gol si mai apoi va trimite mesaje CON , la care clientul va raspunde cu un mesaj ACK.

Daca mesajul de la client este trimis folosind mesaj NON, si raspunsul serverului va fi NON.

Programul nostru suporta mulithreading. Mai multi clienti pot avea acces sa preia temperatura.

Porgramul incepe prin a prelua date despre vremea dintr-un anumit oras cum ar fi in cazul nostru: temperatura (returnata in Kelvin), temperatura minima, temperatura maxima, temperatura resimtita, presiunea si umiditatea. Deoarece temperatura nu este in Celsius, ne-am decis sa transformam temperatura in Celsius creand o noua metoda „CONVERT”.

Am creat un message_header ce contine datele ce mai apoi impreuna cu payloadul (adica data) vor forma un packet. Intrucat odata trimis, pachetul va fi despachetat, ne trebuie o functie de parsare a mesajului, din care extragem toate datele: versiunea, tipul, clasa, codul, message id-ul, tokenul, payloadul si options.

Clientul are rolul de a crea un pachet. Spre exemplu, am creat o lista in care adaugam orase si metoda aplicata pentru un oras (GET, POST, CONVERT). GET are code.class =0.1, Post =0.2, CONVERT=0.6 (celelalte coduri sunt deja luate prin conventie).

Clientul este capabil sa aleaga metoda dorita si in functie de metoda, in pachet adaugam clasa si codul. Versiunea trebuie sa fie 1!

Tokenul este generat de client. Am folosit o functie de generare random a unui token id, precum si a unui message_id. Clientul va trimite pachetul cu aceste token id si message id, iar rolul serverului este de a trimite inapoi clientului cu acelasi token id. Message_id se foloseste pentru a identifica duplicatele.

Pachetul odata ajuns la sever, va fi despachetat si trebuie verificat fiecare element din pachet, iar in caz ca o conditie nu este satisfacuta, se returneaza coduri de eroare (404), iar in caz de succes se returneaza 200/205 (pentru GET).

Versiunea trebuie sa fie 1. Se urmareste ca mesajul sa fie de tip confirmabil, iar daca mesajul este de tip confirmabil, tipul raspunsului va fi ACK. Pentru mesajele NON, nu se asteapta niciun mesaj de raspuns.

In caz ca requestul va fi GET, in urma metodei, clientul (in caz de reusita) va primi un cod CONTENT 2.05 (succes- content), in caz de esec, ERROR 404. Serverul creeaza un pachet in care pune aceste coduri, token-ul id si mid este acelasi pe care l-a primit de la client. In payload se vor pune cele 6 date (eg: temp, umiditate, etc).

In caz ca requestul va fi POST codul va fi 2.01 pentru create si 2.04 in caz de UPDATE. Am stocat intr-un fisier data (eg: <oras> : <temperatura>) . Daca POST nu se realizeaza cu succes, se returneaza mesaj de eroare 404. Se urmareste daca in fisier exista deja orasul, daca nu exista, ii facem update in fisier, iar daca nu exista, il adaugam in fisier.

Metoda CONVERT este simpla. Codul este 0.6, ales deoarece celelalte coduri sunt luate prin conventie de catre celelalte metode. In caz de reusita, codul returna este 200, iar payloadul va fi completat de temperaturile(!) convertite la Celsius.

Bibliografie:

https://support.holmsecurity.com/hc/en-us/articles/212963869-What-is-the-difference-between-TCP-and-UDP-

http://programmingwithreason.com/article-iot-coap.html