Constrained Application Protocol

Server

**Students:**

Chelarașu Elena-Denisa, 1308B

Miron Alexandru, 1308B

**Professor:**

Nicolae-Alexandru Botezatu

**Introducere**

CoAP este un protocol de nivel de serviciu, folosit pentru comunicația eficientă între dispozitive cu resurse limitate, sau între dispozitive și internet. Protocolul are o structură simplă, overhead mic, și are suport pentru multicast. Este un protocol util pentru comunicația machine-to-machine și Internet of Things (IoT).

CoAP are la bază modelul client-server. Clientul transmite cereri serverului, iar serverul le procesează și trimite mesaje de răspuns. Tipurile de cereri ce pot fi transmise sunt asemănătoare cu cele de la HTTP: GET, POST, PUT, DELETE, PATCH, etc.

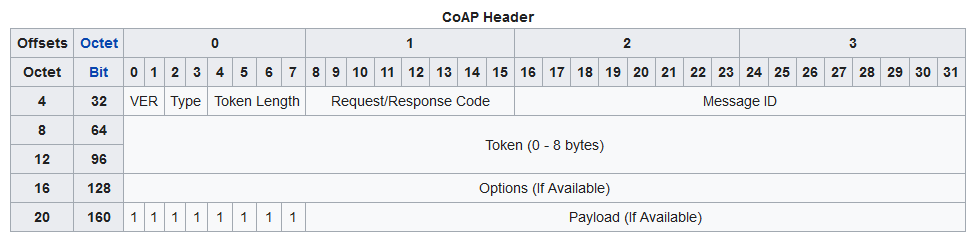
CoAP folosește datagrame UDP pentru mesaje, opțional folosind și DTLS pentru securitatea comunicațiilor. CoAP poate rula pe majoritatea dispozitivelor care au suport pentru UDP (sau ceva similar).

User Datagram Protocol (UDP) este un membrul de bază al suitei de protocoale Internet. Datagramele UDP pot fi trimise fără a stabili o conexiune explicită între sursă și destinatar, însă nu oferă garanții ca protecția contra mesajelor duplicate, pierderea mesajelor, sau sosirea lor în ordine inversă.

Pentru a soluționa această problemă, CoAP definește cereri confirmabile și non-confirmabile, și răspunsuri de tip Acknowledgement și Reset. La recepționarea unei cereri confirmabile, serverul trebuie să răspundă cu un mesaj de tip Acknowledgement dacă a procesat cererea cu succes, sau Reset dacă nu. CoAP folosește un identificator de mesaj pentru detecția mesajelor duplicate și corelarea mesajelor cerere / răspuns, și definește o serie de opțiuni stocate în pachet într-un format compact pentru parametrizarea ușoară a comunicației.

CoAP are suport pentru Proxy și operațiuni de caching prin intermediul unor opțiuni ca Proxy-Uri, Proxy-Scheme, Max-Age, ETag, etc.

**Descriere pachet CoAP**

Un pachet CoAP are o lungime minimă de 4 octeți, ce include următoarele câmpuri:

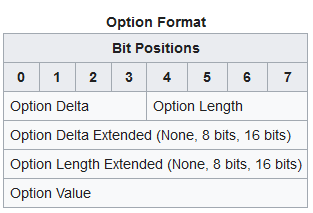
* **Version** - are valoarea constant 01. Alte valori ale acestui câmp sunt rezervate pentru versiuni noi CoAP
* **Type** - determină tipul pachetului
  + 00 - Pachet confirmabil - Destinatarul trebuie să confirme recepționarea mesajului prin ACK sau RESET
  + 01 - Pachet non-confirmabil - Nu este necesară confirmarea, însă destinatarul poate răspunde cu un pachet confirmabil sau non-confirmabil
  + 10 - Pachet de tip Acknowledgement - Indică recepționarea cu succes a mesajului
  + 11 - Pachet de tip Reset - Indică recepționarea unui mesaj a cărui parsare a eșuat
* **Token Length** - determină lungimea în octeți a câmpului Token (0 - 8 octeți)
* **Request / Response Code** - determină tipul mesajului. Primii trei biți indică clasa mesajului, iar ultimii cinci indică codul (spre exemplu, 0.01 reprezintă requestul GET)
* **Message ID** - un identificator folosit pentru detecția mesajelor duplicate și corelarea mesajelor Acknowledgement / Reset la mesajele Confirmabile / Non-confirmabile.

Ulterior, mesajul poate conține următoarele câmpuri, cu o lungime variabilă:

* **Token** - are lungimea în octeți egală cu Token Length. Tokenul reprezintă o valoare generată de client ce trebuie să se regăsească fără modificări în răspunsurile serverului. Tokenul poate fi folosit pentru stabilirea contextului sau stării unui mesaj
* **Options** - stochează un șir de opțiuni într-un format compact, bazat pe reprezentări variabile în lungime a identificatorului opțiunii și a lungimii în octeți.
* **Payload** - este opțional, fiind marcat cu un octet 0xFF (Payload Marker) în locul unde în mod normal ar apărea o opțiune CoAP. Toți octeții rămași din mesaj care sunt după Payload Marker sunt considerați ca făcând parte din Payload.

**Detalii despre câmpurile pachetului**

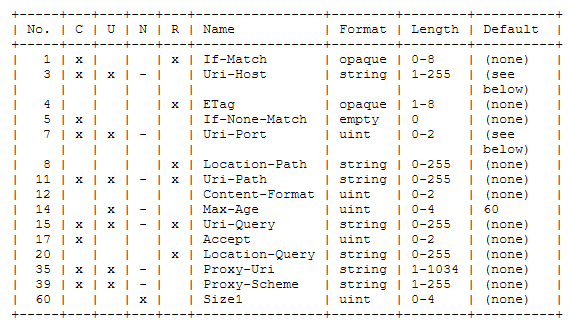
**Options**

Câmpul Options conține mai multe opțiuni pentru comunicația CoAP. IDul opțiunii și lungimea ei sunt codificate în format variabil. IDul opțiunii curente se calculează ca o sumă între IDul opțiunii precedente, și valoarea delta calculată curent.

Valoarea delta, precum și lungimea, sunt codificate în format variabil:

* Pentru Option Delta între 0 și 12, Option Delta Extended are lungimea 0, iar delta este egal cu Option Delta.
* Pentru Option Delta = 13, Option Delta Extended are lungimea de 1 octet, iar delta este egal cu Option Delta Extended - 13
* Pentru Option Delta = 14, Option Delta Extended are lungimea de 2 octeți, iar delta este egal cu Option Delta Extended - 269
* Option Delta = 15 (0xF) este folosit pentru Payload Marker.

Similar se procedează și pentru Option Length.



Opțiunile au următorul rol:

* Uri-Host, Uri-Port, Uri-Path și Uri-Query sunt folosite pentru a formula un URI ce reprezintă resursa dorită
* Proxy-Uri și Proxy-Scheme sunt folosite în cadrul redirectării comunicației printr-un proxy
* Content-Format indică tipul de reprezentare a datelor din Payload. În cadrul aplicației server CoAP, ne propunem tratarea tipurilor 0 (text/plain; charset=url-8) și 50 (application/json).
* Accept indică Content-Formatul preferat de către client pentru răspunsurile serverului. Dacă serverul nu poate returna formatul preferat de client, eroarea 4.06 ”Not Acceptable” trebuie să fie returnată
* Max-Age indică timpul maxim (în secunde) pentru care un request este valid. Opțiunea este folosită pentru operațiile de caching, și indică durata pentru care un răspuns este valid de la trimiterea de către server. Dacă nu este specificată opțiunea, se folosește o valoare implicită de 60 de secunde.
* ETag funcționează ca un ID pentru resurse. ETagurile sunt generate de către server la cererea resurselor de către clienți. Într-un răspuns, ETagul oferă IDul curent al resursei, iar într-o cerere, indică dorința de a valida resursa.
* Location-Path și Location-Query sunt folosite în mesajele de răspuns pentru a indica unde a fost creată o resursă în urma unor operațiuni de tip POST
* If-Match este folosit pentru a realiza updateuri asupra unor resurse existente. Un If-Match urmat de un ETag va acționa doar asupra unei resurse cu acel tag, în timp ce un If-Match gol ar acționa asupra oricărei resurse existente.
* If-None-Match este similar cu If-Match, însă nu are o valoare, și acționează doar în cazul lipsei unei resurse.
* Size1 este folosit în răspunsurl 4.13 pentru a indica dimensiunea maximă acceptabilă de către server a requesturilor.

Opțiunile sunt de mai multe tipuri:

* Critical / Elective - definește comportamentul pentru opțiuni nerecunoscute. Pentru cele de tip ”Critical” serverul va răspunde cu un mesaj de tip ”Bad Option”, iar clientul va respinge mesajul cu opțiunea nerecunoscută. Opțiunile nerecunoscute de tip ”Elective” vor fi ignorate silențios
* Proxy Unsafe / Safe-to-Forward - definește comportamentul pentru opțiuni nerecunoscute, însă în cadrul tratării de către un Proxy-Scheme
* NoCacheKey - este de asemenea folosit de către Proxy

**Payload**

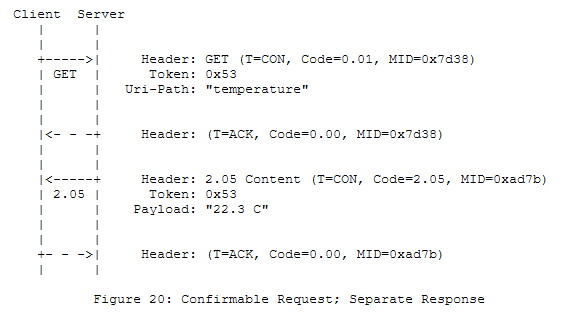
Câmpul Payload de obicei conține reprezentarea unor resurse. În cazul requesturilor de tip POST, PUT, Payload ar conține reprezentarea resursei ce trebuie creată sau modificată. În cazul răspunsurilor la cereri de tip GET, serverul va stoca reprezentarea resursei existente, sau în cazul la POST, PUT, reprezentarea resursei modificate sau a operației făcute.

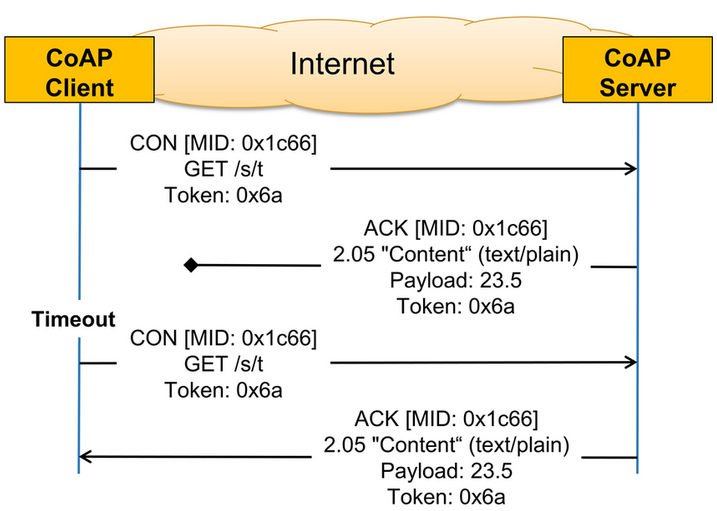
În cazul mesajelor de tip Client Error / Server Error, Payload va conține un șir de caractere în format UTF-8, ce ar explica cauza apariției erorii într-un format ușor de înțeles (spre exemplu, ”The server was unable to complete the request due to an operation timeout”). Aceste mesaje sunt folosite pentru depanarea erorilor.

Unele tipuri de mesaje nu pot conține Payload-uri, iar serverul / clientul trebuie să trateze apariția lui drept o eroare.

**Exemple de comunicații CoAP**

Răspuns într-un mesaj separat (vom utiliza pentru metoda SEARCH)

Retransmiterea cererilor de tip Confirmable

****

**Scopul proiectului**

Obiectivul primar al acestui proiect este implementarea unui server CoAP capabil să comunice cu unul sau mai mulți clienți CoAP. Clienții vor face cereri spre resursele de filesystem a serverului, iar serverul va răspunde cu informații despre fișierele disponibile, sau va executa unele operațiuni precum crearea, ștergerea, redenumirea fișierelor, etc.

Un alt obiectiv este colaborarea cu echipa ce implementează clientul CoAP. Împreună, vom decide formatul mesajelor trimise, formatul datelor trimise, operațiile FS ce vor fi puse la dispoziție, precum și vom testa aplicațiile una contra celeilalte.

**Obiective**

* Implementarea serverului CoAP
* Implementarea operațiilor de tip browser FS
* Comunicarea cu mai mulți clienți CoAP în același timp
* Implementarea codurilor de răspuns CoAP aferente serverului
* Implementarea pe deplin a opțiunilor CoAP, etc.

**Tehnologii folosite**

Aplicația va fi realizată în Python 3, cu ajutorul IDEului PyCharm de la JetBrains.

Ca module adiționale, vom folosi **sockets** pentru primirea și trimiterea pachetelor UDP, **tkinter** pentru interfața grafică, precum și **os** pentru operațiile FS.

**Operații filesystem suportate**

Ne propunem să suportăm următoarele operațiuni pe filesystem:

* cd - Schimbă directorul curent
* dir\_back - Navigare un director înapoi
* list - Listează fișierele din directorul curent
* create\_file - Crează un fișier gol
* create\_dir - Crează un director gol
* rename - Redenumește un fișier sau director
* move - Mută un fișier sau director
* delete - Șterge un fișier sau director
* open - Citește un fișier
* save - Salvează un fișier

Codurile relevante ce le vom implementa vor fi 0.00 (EMPTY), 0.01 (GET), 0.02 (Post), și codurile de răspuns aferente serverului. În plus, vom expanda metodele CoAP prin metoda nouă 0.08 (SEARCH), care va realiza o căutare din directorul curent și va returna fișierele / directoarele găsite.

Pentru metoda SEARCH, este posibil ca durata de căutare să fie mai mare decât timpul acceptabil pentru a returna un răspuns în mesajul Acknowledge. Pentru a rezolva această situație, vom trimite un răspuns ACK gol, și ulterior vom transmite un răspuns confirmabil pentru a transmite rezultatul căutării.

Comenzile vor fi transmise prin componenta Payload a pachetelor CoAP, în format JSON. Structura specifică a datelor JSON urmează să fie stabilită mai târziu de către ambele echipe.

**Detalii de implementare pentru server**

Comunicația client-server va fi logată într-un fișier

Pentru a asigura siguranța datelor locale, vom restricționa accesul clienților la filesystem. Astfel, clienții vor putea opera doar cu fișierele locate într-un director ”root” (similar cu htdocs la un server HTTP Apache).

**Bibliografie**

[RFC 7252](https://www.rfc-editor.org/rfc/rfc7252.html)

[Wikipedia: Constrained Application Protocol](https://en.wikipedia.org/wiki/Constrained_Application_Protocol" \l "Option)

[Python Sockets Module Documentation](https://docs.python.org/3/library/socket.html)

[RFC 7252 The Constrained Application Protocol (CoAP). 12.2. CoAP Option Numbers Registry](https://www.rfc-editor.org/rfc/rfc7252" \l "section-12.2)

[Example of CoAP Confirmable Message (CON) exchange](https://www.researchgate.net/figure/Example-of-CoAP-Confirmable-Message-CON-exchange-If-the-client-does-not-receive-an-ACK_fig2_305522152)