Universitatea Tehnică “Gheorghe Asachi” din Iaşi

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

REȚELE DE CALCULATOARE

proiect

TEMA : Client CoAP

Studenţi:
Epure Andrei-Ioan
Lungu Bogdan-Andrei

Grupa :
1309A

Coordonator:

Nicolae-Alexandru Botezatu

2022

Constrained Application Protocol (CoAP)

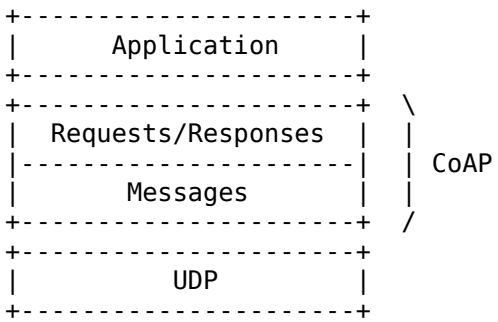
Constrained Application Protocol (CoAP) este un protocol specializat de transfer pentru utilizarea cu noduri si retele constranse (de exemplu ,consum redus de energie,cu pierderi).
Protocolul este conceput pentru aplicatii machine-to-machine(M2M),cum ar fi energia inteligenta si automatizarea cladirilor.

Caracteristici:

- Protocol web care indeplineste cerintele M2M in medii constranse
- Schimburi de mesaje asincron
- URI si Content-Type support
- Capabilitati de proxy si caching

CoAP ofera un model de interactiune cerere/raspuns intre punctele finale ale aplicatiei.Acest protocol este conceput pentru a interfata cu usurinata cu HTTP pentru integrarea cu web in timp ce indeplineste cerinte specializate cum ar fi suportul multicast, cheltuielile reduse si simplitate pentru medii constranse.

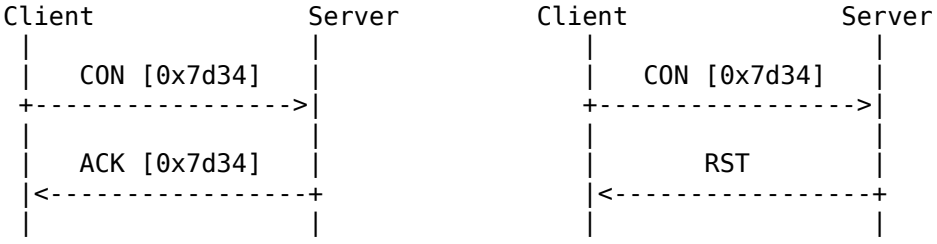
Stratificare abstracta a CoAP



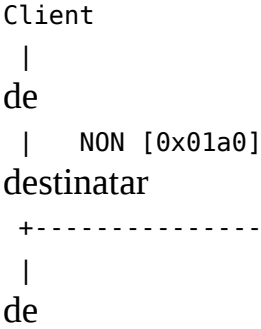
In acest protocul sunt definite 4 tipuri de mesaje:

-Confirmabile(CON)

Un mesaj confirmabil este retransmis folosind un timeout implicit si o retragere exponentiala intre retransimisii,pana cand destinatarul trimite un mesaj de confirmare(ACK) cu acelasi ID.Cand destinatarul nu e capabil sa proceseze un mesaj confirmabil, acesta raspunde cu un mesaj de reset(RST) in loc de confirmare (ACK).



-Non-confirmabile(NON)

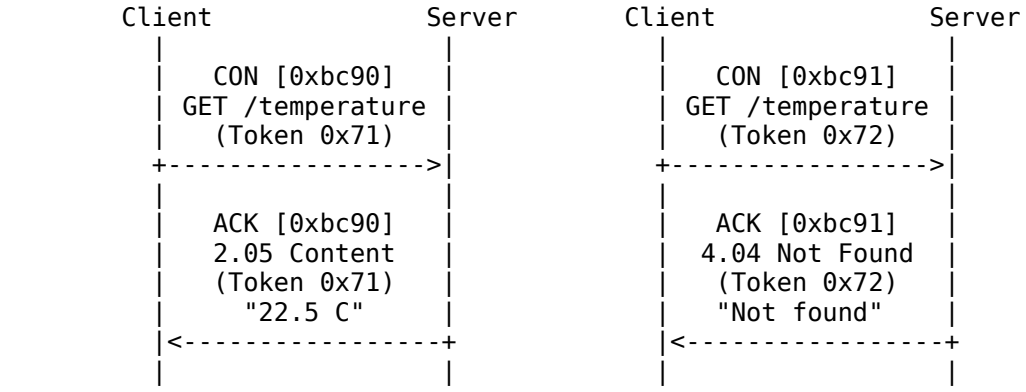


Acestea nu sunt recunoscute ,dar au inca un ID de mesaj pentru detectarea dublelor.Cand un destinatar nu este capabil sa proceseze un mesaj neconfirmabil ,acesta poate raspunde cu un mesaj de resetare

-Acknowledge(ACK) : confirma faptul ca mesajul trimis a ajuns la destinatie

-Reset(RST): indica ca mesajul a fost primit,dar nu poate fi procesat in mod corespunzator datorita unor probleme

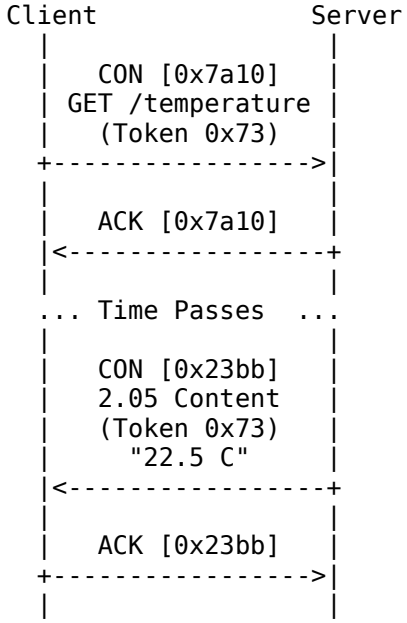
Modelul cerere/raspuns



Exemplu de cerere si returnarea cu succes al raspunsului

Exemplu de cerere si returnarea a unui cod de erore

Daca serverul nu este capabil sa raspunda imediat unei cereri atunci se trimite un mesaj confirmabil cu acesta iar clientul va trimite un acknowledge catre server cu scopul de a confirma server-ului ca a primit raspunsul.



Structura unui mesaj CoAP

CoAP Message

Octet offset		0								1								2								3							
	Bit offset	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)																							

Ver- 2 biti,indica numarul versiunii CoAP.Implementarile acestei specificatii TREBUIE sa seteze acest camp la 1 (01 binar)

Type – 2 biti, indica daca mesajul e de tipul Confirmable(0),Non-Confirmable(1),Acknowledge(2) or Reset(3)

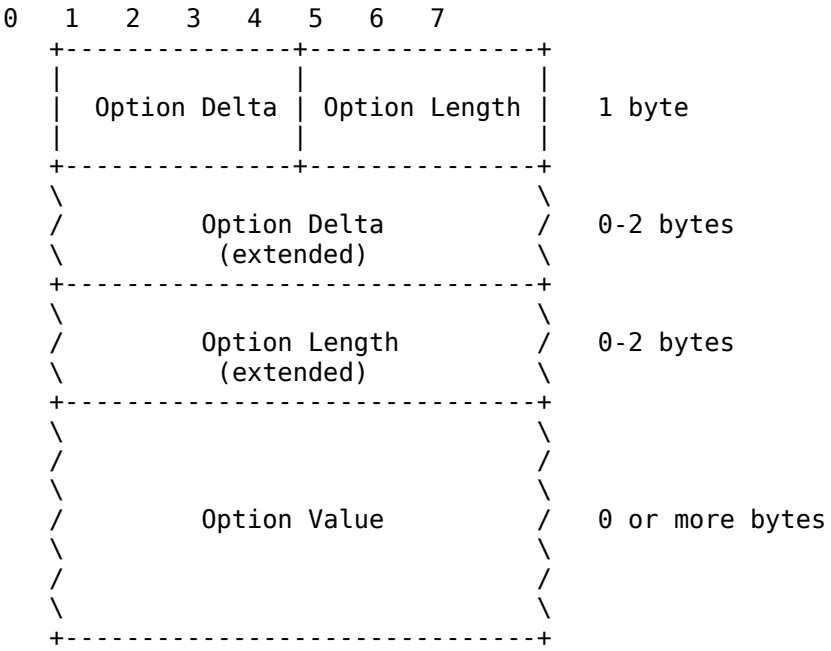
Token length – 4 biti,indica lungimea campului token de lungime variabila (0-8 octeti).Lungimile 9 –15 sunt rezervate,NU TREBUIE trimise si TREBUIE procesate ca mesaj de eroare.

Code – 8 biti,impartiti in 3 biti clasa(cei mai semnificativi),5 biti detaliu(cei mai putin semnificativi). Format : c.dd ,c este o cifra intre 0 si 7 ,dd este intre 00-31.Clasa poate indica un request(0),un raspuns de succes(2),un raspuns de eroare client(4) sau eroare de server(4).

ID-16 biti,folosit pentru a detecta duplicarea mesajelor si pentru a potrivi mesajele de timp Acknowledge/Reset cu mesaje de tip Confirmable/Non-Confirmable.

Token – poate fi intre 0 si 8 octeti,asa cum e data de campul Token length.Valoarea token-ului este folosita pentru a corela cererile si raspunsurile.Clientul Trebuie sa genereze token-uri in asa fel incat token-urile utilizate în prezent pentru o anumită pereche de puncte finale sursă/destinație să fie unice.
Un mesaj gol are câmpul Code setat la 0.00. Câmpul Token Length TREBUIE să fie setat la 0 și octeții de date NU TREBUIE să fie prezenți după câmpul ID . Dacă există octeți, aceștia TREBUIE să fie procesați ca o eroare de format de mesaj.

Option format



Option Delta: întreg fara semn pe 4 biți. O valoare între 0 și 12 indică opțiunea Delta.
Trei valori sunt rezervate :

13: Un întreg fără semn de 8 biți urmează octetul inițial și indică opțiunea Delta minus 13.

14: Un număr întreg fara semn pe 16 biți în ordinea octeților de rețea urmeaza octetul inițial și indică opțiunea Delta minus 269.

15: Rezervat pentru marcatorul de sarcină utilă. Dacă câmpul este setat la această valoare, dar întregul octet nu este marcatorul de sarcină utilă, acest lucru TREBUIE să fie procesată ca o eroare de format de mesaj.

Option Length: întreg fără semn pe 4 biți. O valoare între 0 și 12 indică lungimea valorii opțiunii, în octeți.
Trei valori sunt rezervate :

13: Un întreg fără semn pe 8 biți precede Valoarea Opțiunii și indică lungimea opțiunii minus 13.

14: Un întreg fara semn pe 16 biți în ordinea octeților de rețea precede Valoarea opțiunii și indică Lungimea opțiunii minus 269.

15: Rezervat pentru utilizare ulterioară. Dacă câmpul este setat la această valoare,

TREBUIE procesat ca o eroare de format de mesaj.

Valoare: o secvență de octeți data de Option length. Lungimea și formatul valorii opțiunii depind de opțiunea respectivă, care POATE defini valori cu lungime variabilă.

Definirea pachetelor

1. Operatia de citire a ierarhiei unui director (code 0.01)

Exemplu structura payload:

{
"op": "ls",
"path": "/"
}

2. Operatia de citire a unui fisier (code 0.01)

Exemplu structura payload:

{
"op": "cat",
"path": "/director/fisier"
}

3. Operatia de creare a unui fisier / director (code 0.02)

Exemplu structura payload:

{
"op": "touch",
"path": "/director/fisier",
"idx": 1 ,
"total": 3
}

Observatie: in campul data se vor transmite octetii corespunzatori indexului (field-ul "idx"), necesar la ordonarea octetilor transmisi.

Exemplu structura payload:

{
"op": "mkdir",
"path": "/director"
}

4. Operatia de mutare a unui fisier (code 0.03)

Exemplu structura payload:

{
"op": "mv",
"path": "/director/fisier",
"newPath": "/fisier"
}

5. Operatia de stergere a unui fisier / director (code 0.04)

Exemplu structura payload:

{
"op": "rm",
"path": "/director/fisier"
}

6. Operatia de cautare a unei secvente text in fisierele dintr-un director (code 0.01)

Exemplu structura payload:

{
"op": "find",
"path": "/director"
"search": "secventa"
}

Method Codes

Code	Name	Reference
0.01	GET	[RFC7252]
0.02	POST	[RFC7252]
0.03	PUT	[RFC7252]
0.04	DELETE	[RFC7252]

0.00 -> Mesaj gol

Request/response code (8 bits)

```
0 1 2 3 4 5 6 7
```

Format este c.dd,unde “c” reprezinta clasa,iar “dd”

```
+-+-+-+-+-+-+-+-+
|class|  detail |
+-+-+-+-+-+-+-+-+
```

reprezinta detaliul
Exemplu : “Not Found” este scris sub forma 4.04

Exista 3 clase de coduri de raspuns :

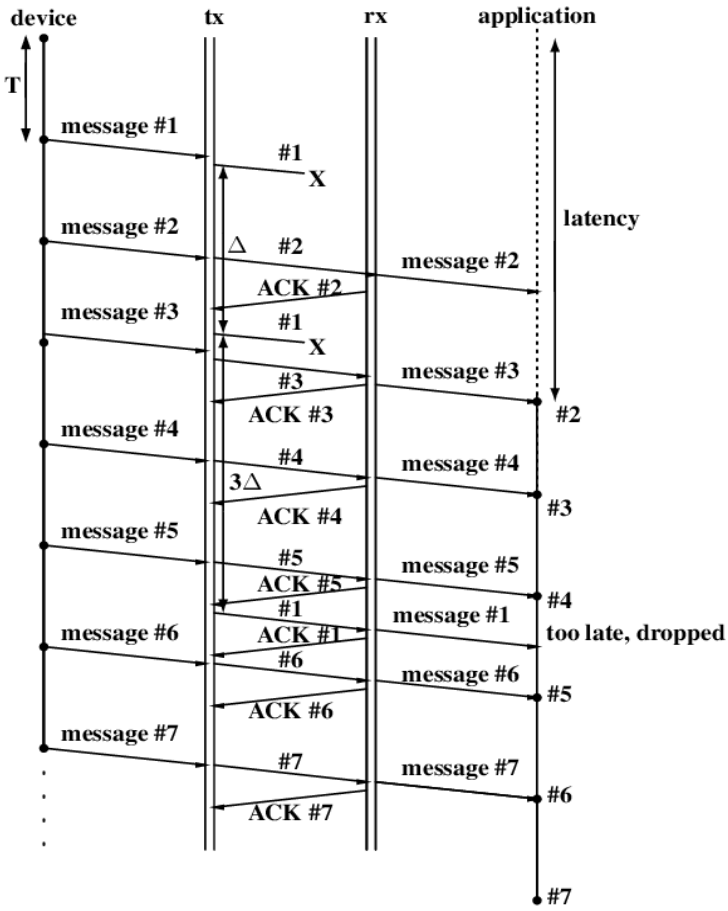
2 – Success: cererea a fost cu succes primita,inteleasa si acceptata
4 – Client Error : cererea contine erori de sintaxa sau nu poate fi indeplinita
5 – Server Error : serverul a esuat in a indeplini o cerere valida

User Datagram Protocol (UDP)

În rețelele de calculatoare, User Datagram Protocol (UDP) este unul dintre protocoalele de comunicare de bază ale suitei de protocoale Internet utilizate pentru a trimite mesaje (transportate ca datagrame în pachete) către alte gazde dintr-o rețea Internet Protocol (IP). În cadrul unei rețele IP, UDP nu necesită o comunicare prealabilă pentru a configura canale de comunicare sau căi de date.

Caracteristici ale UDP ce il face potrivit pentru anumite aplicatii:
- Este orientat către tranzacții, potrivit pentru protocoale simple de interogare-răspuns,precum DNS sau NTP
- Lipsa întârzierilor de retransmisie îl face potrivit pentru aplicații în timp real
- Potrivit pentru un număr foarte mare de clienți, cum ar fi aplicațiile media de streaming precum IPTV

Transactions of CoAP over UDP



Structura unui pachet UDP:

1. Pseudo-antetul IPV4

Biți	0 - 7	8 - 15	16 - 23	24 - 31
0	Adresa sursa			
32	Adresa destinație			
64	Zero	Protocol	Lungime UDP	

Observatie: Campul Protocol are valoarea 17 (hexazecimal 0x11) specifica protocolului UDP.

2. Antetul

Biti	0 - 15	16 - 32
0	Portul sursa	Portul destinație
32	Lungime	Suma de control
64	Date	

- **Portul sursa** - în adresarea bazata pe IPv4 acest câmp este opțional. Daca nu este utilizat acest câmp, are valoarea zero; când reprezinta informație semnificativa, el va indica portul inițiator al procesului de transmisie a datagramelor.
- **Portul destinație** - spre deosebire de portul sursa, câmpul este obligatoriu și indica portul de recepție
- **Lungime** - acest câmp indica lungimea în octeți a datagramei: antet plus secțiune de date (valoarea minima a lungimii este 8).
- **Suma de control** - asigura imunitatea la erori; se calculează ca fiind complementul fata de 1 (pe 16 biți) a pseudo-antetului cu informații extrase din antetul IP, antetului UDP și a câmpului de date, eventual se completează cu zerouri pentru a atinge lungimea stabilita.

Observatie: Ca urmare a limitarii pachetelor transmise folosind IPV4 se pot transmite maxim 65.507 octeti.

Particularitatile protocolului UDP:

- Conceptele de ACK, RST sau timeout nu sunt specifice acestui protocol.
- Ordinea primirii pachetelor nu corespunde neaparat cu ordinea trimiterii.
- Intrucat mecanismele mentionate aterior lipsesc, procesarea pachetelor este eficienta.