Création d'Applications Internet

TP05 - extension du service RESTful et de l'application client afin de réaliser l'observation des données météo

Patrick Audriaz & Bruno Tschopp

Prof. Serge Ayer

Table des matières

1. Introduction

L'objectif de ce TP à été d'ajouter la fonctionnalité d'observation du service, afin de supprimer les requêtes inutiles vers le serveur en mettant en place une système de notification bidirectionnel.

2. Questions

2.1 P1 - Analyse de trafic. A l'aide d'un outil d'analyse de trafic, affichez le trafic réseau pour la requête HTTP représentant la requête EventSource, ainsi que pour le paquet TCP contenant une des données (par ex. la température) mise à jour après chaque observation.

En cliquant sur la requête GET depuis la Loopback, nous avons utilisé l'outil "Suivre flux TCP". Nous avons pu voir des informations concernant la requête et sa réponse.

```
GET /sse/coap://appint02.tic.heia-fr.ch HTTP/1.1
Host: localhost:8586
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/event-stream
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Origin: null
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache

HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Origin, X-Requested-With, Content-Type, Accept
Access-Control-Allow-Methods: PUT, POST, GET, DELETE, OPTIONS
Content-Type: text/event-stream
Cache-Control: no-cache
Connection: keep-alive
Date: Mon, 13 May 2019 14:30:36 GMT
Transfer-Encoding: chunked
```

FIGURE 2.1 – Requête et réponse HTTP pour la méthode GET

Nous avons également pu y voir le flux TCP avec suivant la fréquence défini par le ping. Nous pouvons voir ci-dessous que des données ne sont pas toujours présentes.

```
24c
data: {"temperature":{"name":"Current temperature","description":"Ambient temperature measured at floor level.","unit":"celsius","current_condition":{"value":11.729999999999999,"date":"13.4.2019","time":"14:30:37"}},"humidity":
{"name":"Current humidity","description":"Ambient humidity measured at floor level.","unit":"%","current_condition":{"value":50.73,"date":"13.4.2019","time":"14:30:37"}},"pressure":{"name":"Current pressure","description":"Atmospheric
pressure measured at floor level.","unit":"hPa","current_condition":{"value":1027.55,"date":"13.4.2019","time":"14:30:37"}}}


8
data:


8
data:


6
id: 1

23f
data: {"temperature":{"name":"Current temperature","description":"Ambient temperature measured at floor level.","unit":"celsius","current_condition":{"value":12.53,"date":"13.4.2019","time":"14:30:47"}},"humidity":{"name":"Current
humidity","description":"Ambient humidity measured at floor level.","unit":"%","current_condition":{"value":50.32,"date":"13.4.2019","time":"14:30:47"}},"pressure":{"name":"Current pressure","description":"Atmospheric pressure measured
at floor level.","unit":"hPa","current_condition":{"value":1026.45,"date":"13.4.2019","time":"14:30:47"}}}
```

FIGURE 2.2 – Flux de données TCP

2.2 P2 - Analyse de trafic. A l'aide d'un outil d'analyse de trafic, affichez le trafic réseau (couche COAP – entre le serveur HTTP/Proxy et le serveur COAP) requis pour une observation de la donnée analysée dans la réponse à la question 1.

Lorsque nous avons observer le trafic COAP entre notre serveur HTTP/Proxy et le serveur COAP. Nous avons vu qu'il y avait trois types de messages différents avec COAP.

Un premier message annonçant au serveur que le client vas "listen". Le serveur devra lui communiquer les évènements à venir. Le message numéro 965 est donc envoyé par le client vers le serveur.

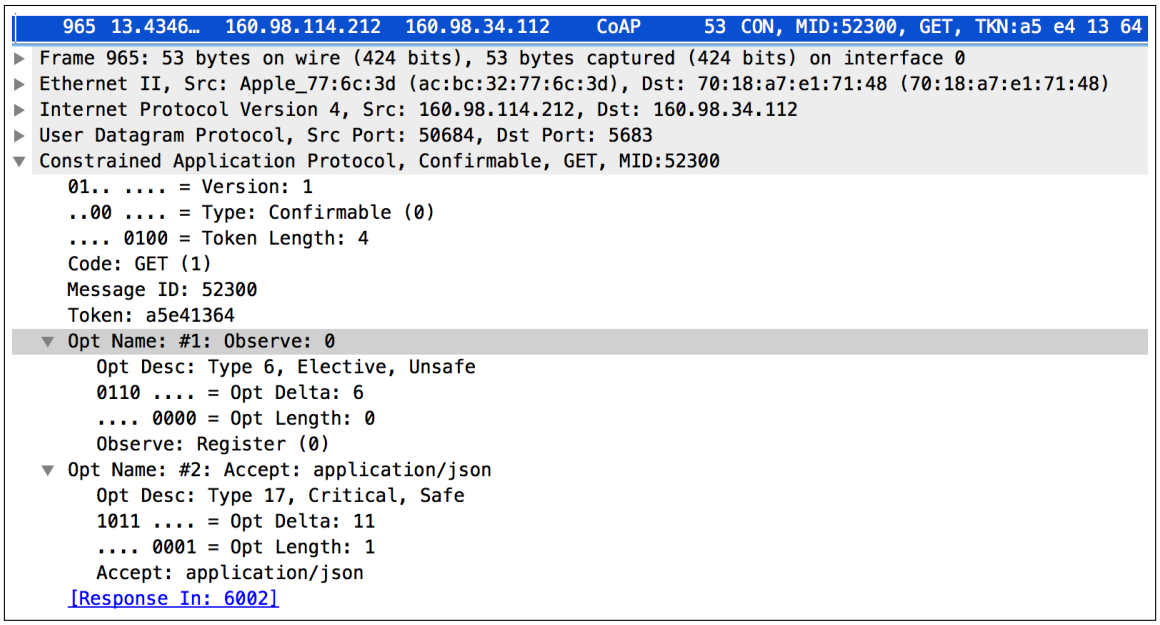

FIGURE 2.3 – Le client s'annonce comme listener sur le serveur

Nous pouvons voir ci-dessous que le serveur va lui répondre avec les données plus tard, le champ [Request In : 965] nous indique que l'envoi des données fais suite à l'annonce de "listen" émise plus tôt.

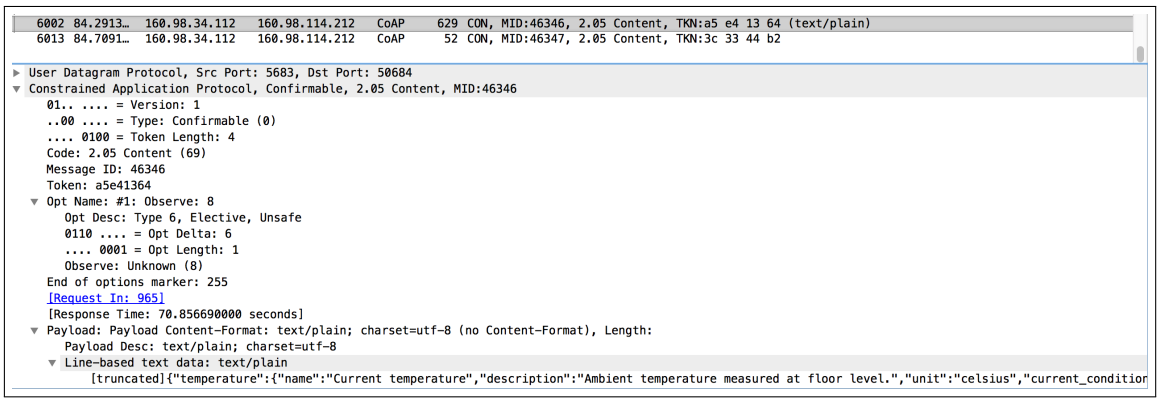

FIGURE 2.4 – Diffusion des données au près du client

Ce troisième type de message est envoyer par le serveur COAP au client pour lui confirmer que la connexion est toujours valable. Le client vas donc continuer de "Listener" bien qu'il n'y ai pas eu d'évent récent.

```
6013 84.7091…  160.98.34.112   160.98.114.212   CoAP      52 CON, MID:46347, 2.05 Content, TKN:3c 33 44 b2
▶ Frame 6013: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface 0
▶ Ethernet II, Src: 70:18:a7:e1:71:48 (70:18:a7:e1:71:48), Dst: Apple_77:6c:3d (ac:bc:32:77:6c:3d)
▶ Internet Protocol Version 4, Src: 160.98.34.112, Dst: 160.98.114.212
▶ User Datagram Protocol, Src Port: 5683, Dst Port: 57545
▼ Constrained Application Protocol, Confirmable, 2.05 Content, MID:46347
    01.. .... = Version: 1
    ..00 .... = Type: Confirmable (0)
    .... 0100 = Token Length: 4
    Code: 2.05 Content (69)
    Message ID: 46347
    Token: 3c3344b2
  ▼ Opt Name: #1: Observe: 5
      Opt Desc: Type 6, Elective, Unsafe
      0110 .... = Opt Delta: 6
      .... 0001 = Opt Length: 1
      Observe: Unknown (5)
```

FIGURE 2.5 – Requête et réponse HTTP pour la méthode GET

2.3 P3 - Dans un diagramme, documentez les échanges de données entre les différents acteurs du système pour la requête EventSource et pour chaque observation de données. La documentation doit se faire pour les couches analysées dans les réponses aux questions 1 et 2.

Le client vas lancer une requête de connexion que sera relayé par le notre serveur coap.js vers le device.

Lorsque de nouvel données seront récoltées par le device le serveur proxy sera notifiés. Il va ensuite envoyé les données vers le client. Dans le cas ou aucune nouvel données apparaît durant une certaine période de temps, un ping sera envoyé par le serveur afin de maintenir la connexion.

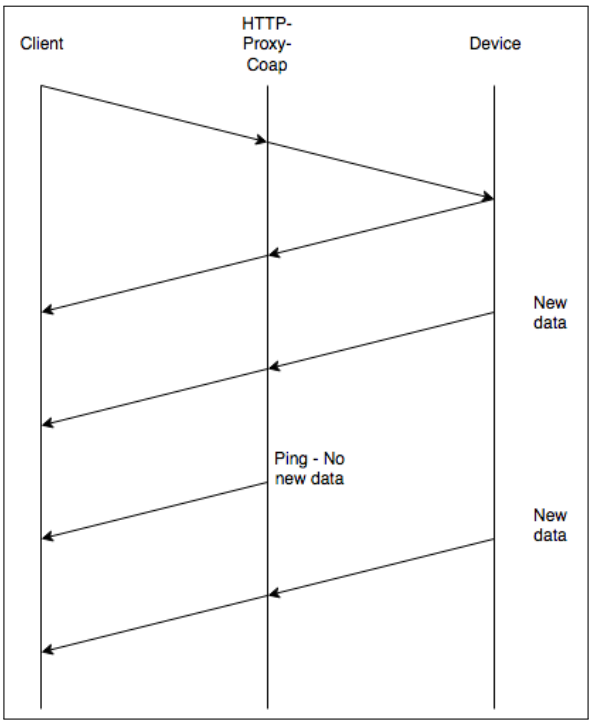

FIGURE 2.6 – Graphique des interactions

3. Conclusions

Nous sommes heureux d'être arrivé à la fin de ce travail intégré. De plus le résultat atteint est satisfaisant pour nous malgré les quelques petits bugs restants.

Fribourg, 15 mai 2019

Patrick Audriaz

Bruno Tschopp