COAP (Constrained Application Protocol) : Protocole d'Application pour l'Internet des Objets

EFORT
http://www.efort.com

1 Introduction

L'IETF a récemment crée un nouveau groupe de travail : Constrained RESTFul Environment CORE. L'objectif de ce groupe est d 'étendre l'architecture WEB aux applications Machine to Machine (M2M) et Internet des Objets (IoT, Internet of Things) utilisant des systèmes contraints. Pour cela, le groupe a défini COAP (Constrained Application Protocol), un protocole de communication générique optimisé pour les architectures contraintes (RFC 7252).
Le protocole CoAP est principalement destiné aux équipements et aux machines qui n'ont parfois qu'un microcontrôleur 8 bits pour tout processeur, très peu de mémoire et qui, en prime, sont connectés par des liens radio lents et peu fiables (les « LowPAN » ou les LPWAN), allant parfois à seulement quelques dizaines de kb/s.
Pour de tels équipements, les protocoles comme HTTP et TCP sont trop contraignants. CoAP est un protocole « HTTP-like », i.e., il reprend nombre de termes et de concepts de HTTP, comme le modèle REST avec la mise en place de systèmes de proxy et de cache, mais spécialement conçu pour des applications M2M dans des environnements à fortes limitations matérielles.
La proximité avec HTTP facilite notamment le développement de passerelles entre le monde des objets CoAP et le Web actuel. Une des pistes explorées pour l'Internet des Objets (IdO) avait été de prendre le HTTP normal et de comprimer pour gagner des ressources réseau mais CoAP suit une autre voie : définir un protocole qui est essentiellement un sous-ensemble de HTTP.
Le but de ce tutoriel est de décrire le protocole COAP avec son format de message, des exemples d'interactions basées sur ce protocole, et comment il permet la découverte des capacités d'un devices.

2 Fonctionnement du protocole COAP

COAP est un protocole REST. REST (Representational State Transfer) ou RESTful est un style d'architecture permettant de construire des applications (Web, Intranet, Web Service). Il s'agit d'un ensemble de conventions et de bonnes pratiques à respecter et non d'une technologie à part entière. L'architecture REST utilise les spécifications originelles du protocole HTTP :

- l'URI (Uniform Resource Identifier) comme identifiant des ressources
- les verbes HTTP comme identifiant des opérations
- les réponses HTTP comme représentation des ressources. Il est important d'avoir à l'esprit que la réponse envoyée n'est pas une ressource, c'est la représentation d'une ressource. Ainsi, une ressource peut avoir plusieurs représentations dans des formats divers : HTML, XML, JSON, etc. C'est au client de définir quel format de réponse il souhaite reçevoir via l'entête Accept.

CoAP fonctionne sur UDP. Pour sécuriser les échanges, il est aussi possible d 'utiliser COAP sur DTLS. Mais les messages COAP peuvent aussi être transportés sur SMS, TCP ou SCTP.
CoAP fonctionne de manière asynchrone.
CoAP s 'appuie sur une approche à deux couches, une couche de messagerie CoAP utilisée afin de traiter la non fiabilité d 'UDP ainsi que la nature asynchrone des interactions (4 messages sont définis CON, ACK, NON, RST) et une couche d 'interaction sous forme de requête/réponse héritée du protocole HTTP (Requêtes GET, POST, PUT, DELETE et catégories de réponses).
Toutefois CoAP n 'est qu 'un seul protocole.
La taille du payload d 'un message CoAP ne doit pas dépasser 1024 octets, mais un mécanisme de transfert de bloc de données CoAP permet l 'envoi de différents fragments d 'un même message, chaque fragment était considéré comme un message.
CoAP supporte l 'envoi de messages en multicast.

La sémantique des requêtes et réponses COAP est transportée dans des messages COAP qui incluent soit un code de méthode (GET, POST, DELETE, PUT) ou un code de réponse (e.g., 404, 205, etc). Un jeton (token) est utilisé pour associer une requête à une réponse indépendamment des messages qui les transportent. Le Token est indépendant de l 'identificateur de message.
Une requête est transportée dans un message CON (Confirmable) ou NON (Non-confirmable) et si la réponse est disponible immédiatement dans le cas d 'un message CON, elle est transportée dans un message Acknowledgement (ACK). Si le message ACK est perdu, l 'émetteur du message CON le retransmettra.
Deux exemples sont montrés concernant une requête de base GET transportée dans un message CON avec la réponse retournée dans un message ACK. Une des réponses indique un succès (2.05) et l 'autre indique un échec (4.04)
Si le serveur n 'est pas en mesure de répondre immédiatement à une requête (e.g., GET) transportée dans un message CON, il répond simplement via un message ACK sans contenu. Lorsque la réponse est prête, le serveur l 'émet dans un nouveau message CON (qui devra être acquitté par le client via un message ACK sans contenu). Ce cas est aussi illustré à la Figure 1.

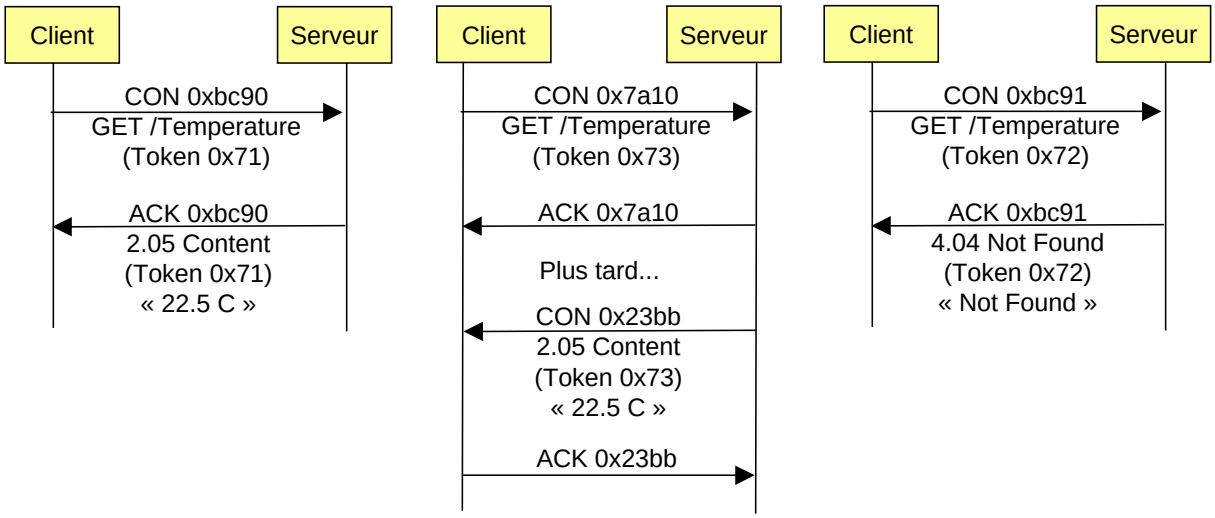

Figure 1 : Messages CON et ACK

Si une requête est émise dans un message Non-confirmable (NON), alors la réponse est émise via un nouveau message NON.
Ce type d 'échange est illustré à la Figure 2.

Lorsqu ’un récepteur n ’est pas du tout en mesure de traiter un message CON (même pas en mesure de fournir une réponse d ’erreur), il répond avec un message Reset (RST) à la place du message Acknowledgement (ACK).
Lorsqu ’un récepteur n ’est pas du tout en mesure de traiter un message NON, il peut répondre avec un message Reset (RST).

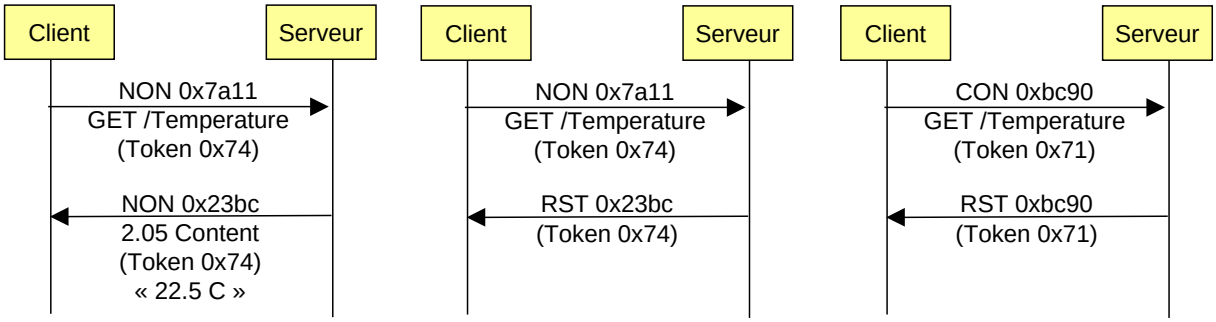

Figure 2 : Message NON

3 Message COAP

L’en-tête CoAP a été conçu pour être facile à analyser par des programmes tournant sur de petits équipements comme les capteurs (Figure 3) . Au lieu du texte lisible de HTTP, l’en-tête de CoAP commence par une partie fixe de quatre octets, qui comprend :La version du protocole (Ver) : il s ’agit de la version du protocole CoAP, à savoir 1.

- Le type de message (T): Indique su le message est de type Confirmable (0), Non-confirmable (1), Acknowledgement (2), ou Reset (3).
- La longueur du Token (TKL): Indique la longueur (variable) du champ Token (0-8 octets) Les longueurs 9-15 sont réservées et ne doivent pas être utilisées. Si un récepteur reçoit une valeur comprise entre 9 et 15, il doit la traiter comme une erreur de format.
- Code: Entier sur 8 bits, séparé en 3 bits de class (bits les plus significatifs) et 5 bits de détail (bits les moins significatifs), documentés sous la forme c.dd ou « c » est un digit compris entre 0 et 7 (3 bits) et « dd » représente deux digits entre 00 et 31 (5 bits). La classe peut indiquer une requête (0), une réponse de succès (2), une réponse d ’erreur client (4) ou une réponse d ’erreur de serveur (5). Toutes les autres valeurs de classe sont réservées. Un cas particulier est celui d ’un message vide dont le code est 0.00. Dans le cas d ’une requête, le champ Code indique la méthode. 0.01 = GET; 0.02 = POST; 0.03 = PUT; 0.04 = DELETE.
- Un Message ID sur deux octets. Ce Message ID permet de détecter les duplicatas et d’associer un accusé de réception à un message précis. À noter qu’avec ces deux octets, on est limité à environ 250 messages par seconde (en raison du paramètre EXCHANGE_LIFETIME qui est à 247 secondes par défaut). Ce n’est pas une limite bien grave : les ressources des machines CoAP ne leur permettent pas d’être trop bavardes de toutes les façons.

L ’en-tête est suivi par la valeur du Token dont la longueur est indiquée par le champ TKL. La valeur de token est utilisée afin de corréler une requête et sa réponse.
L ’en-tête et le Token sont suivis par zero, une ou plusieurs options.
Une option peut être suivie par la fin du message, par une autre option ou un marqueur de payload (contenu) et le contenu lui-même.
Le payload est optionnel. Si présent, il est préfixé par un marqueur de payload d ’un octet (0xFF), qui indique la fin des options et le début du payload. Le payload commence après le marqueur et termine à la fin du datagramme UDP.

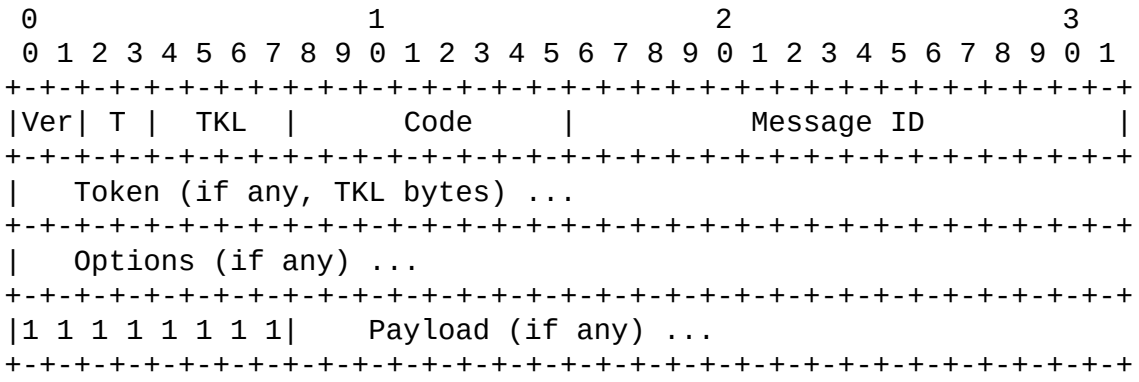

Figure 3 : Format du message COAP

La figure 4 montre une requête GET de base. Le client émet une requête GET dans in message CON au serveur concernant la ressource CoAP://server/temperature, avec un Message ID dont la valeur est 0x7d34. La requête inclut une option Uri-Path (Delta 0 + 11 = 11, Longueur 11, Value "temperature"); le Token est absent. Cette requête a une longueur totale de 16 octets. Une réponse 2.05 (Content) est retournée dans le message ACK qui acquitte la requête émise dans le message CON, répétant le Message ID 0x7d34 et la valeur nulle de Token. La réponse inclut un payload (contenu) qui est "22.3 C" qui a pour longueur 11 octets.

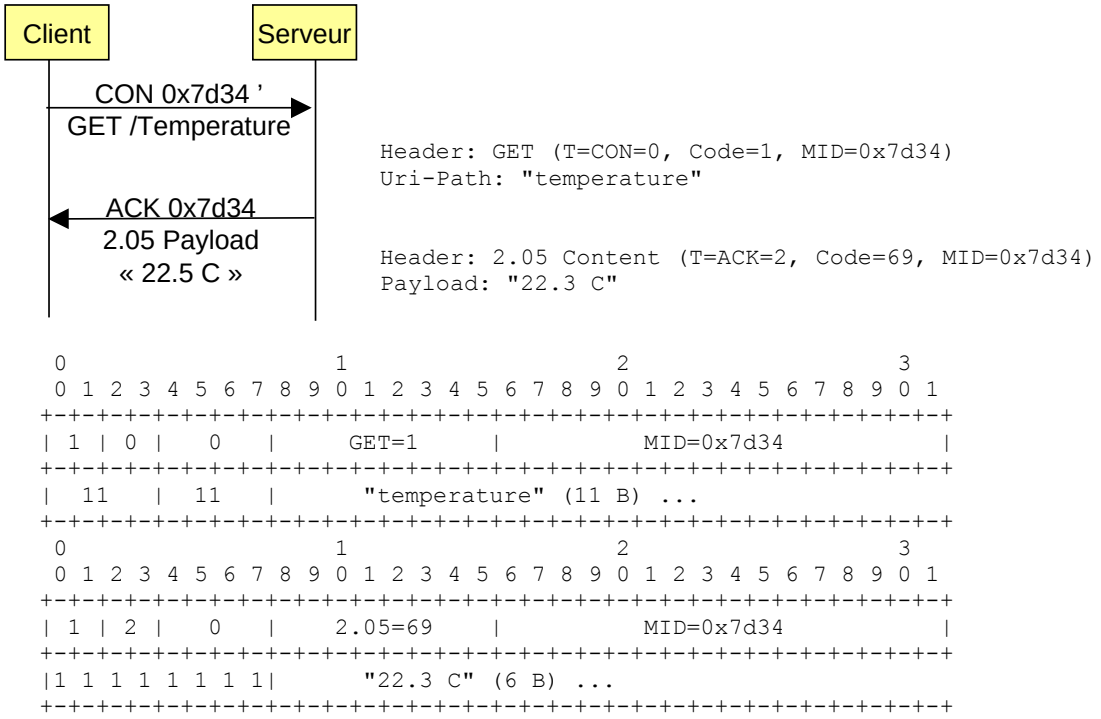

Figure 4 : Exemple de requête et de réponse CoAP

4 Segmentation d'un message CoAP

CoAP s'appuie sur UDO ou DTLS, ce qui limite la taille maximum des représentations des ressources qui peuvent être transférées sans fragmentation. Même si UDP supporte des tailles de segments de données pouvant atteindre 64000 octets en considérant qu'IP assurera la fragmentation, cela ne fonctionne pas réellement pour des applications

contraintes comme celles relatives à M2M. Plutôt que de s'appuyer sur la fragmentation IP, CoAP propose ses propres mécanismes de fragmentation via le transfert des représentation des ressources en utilisant plusieurs blocs de requête/réponse. Cette option permet au serveur dê fonctionner sans état, sans établissement de connexion et sans avoir à mémoriser les précédents transferts de bloc. L'option bloc permet donc de manière minimaliste un transfert de représentation de ressource important.

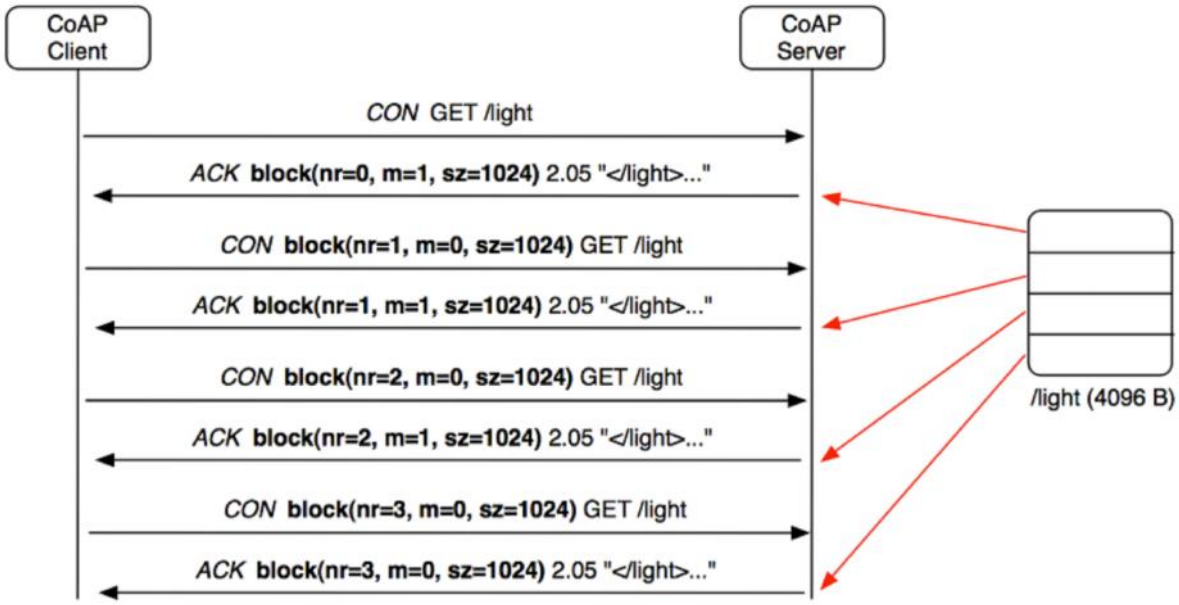

Figure 5 : Segmentation d'un message CoAP

5 Découverte des ressources avec COAP

Afin de découvrir les ressources d 'un serveur COAP, ce dernier doit posséder un chemin commençant par .wellknown/.
Ainsi, un client peut envoyer une requête GET a un serveur avec une URI .wellknown/core et reçoit, en retour une liste de ressources que le serveur propose, e.g., un capteur de température, de luminosité , etc.
Le client peut envoyer une requête en multicast, c'est-à-dire a plusieurs serveurs ou en unicast, a un unique noeud. Si le nombre de noeuds cibles via le multicast est trop important, les réponses des serveurs peuvent surcharger le réseau WPAN. Pour résoudre le problème, un client peut cibler sa recherche de ressources selon des critères passés en paramètre de la requête GET. Il va pouvoir spécifier quel type de ressources il recherche, e.g., /.well-known/core ?rt=Light
Ici, seuls les serveurs possédant une ressource Light doivent répondre a la requête. Les serveurs répondent aux requêtes avec un payload au format suivant :
<$uri-path>;ct=$ct;rt="$rt »
$uri-path designe l'uri que le client doit fournir pour une requête ultérieure; $ct designe les content-types que le serveur connaît; $rt designe le nom de la ressource

6 Interfonctionnement entre CoAP et HTTP

Dans beaucoup de cas, les applications M2M/IoT communiqueront via le protocole HTTP alors que les capteurs/actionneurs supporteront le protocole CoAP compte tenu des

contraintes que leurs sont liés. Dans ce contexte, il est nécessaire de disposer d'un proxy qui disposera en fonction de fonctions de cache pour stocker des informations reçues des capteurs (serveur CoAP) et les mettre à la disposition des applications (client HTTP) tant que ces données sont valeurs (leur max-age n'a pas expiré). La traduction de protocole HTTP/CoAP réalisée par le proxy est relativement simple car les deux protocoles utilisent les mêmes commandes.

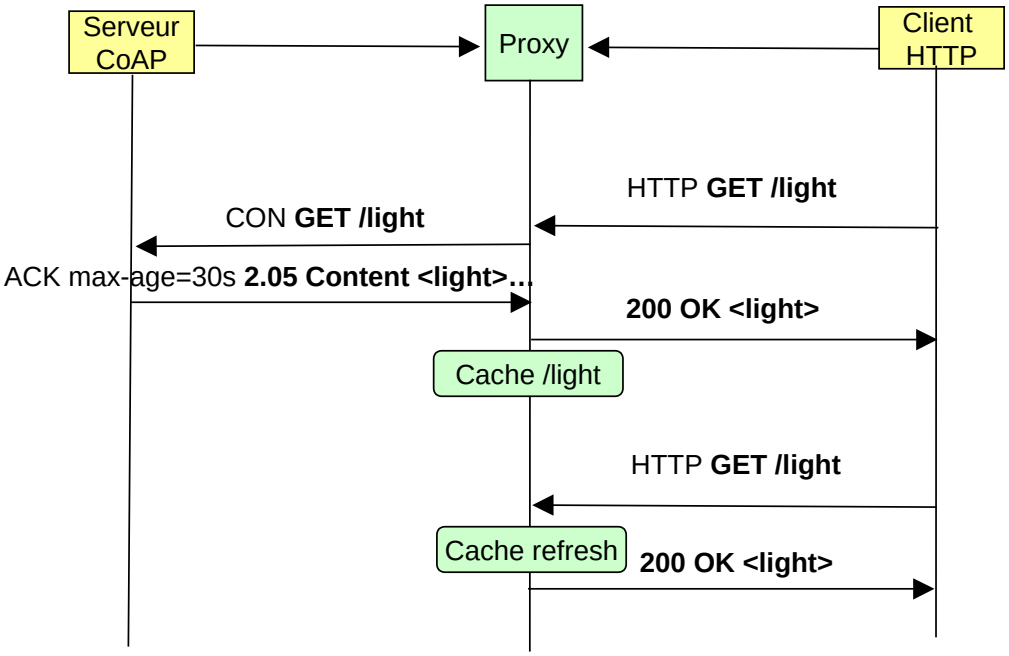

Figure 6 : Interfonctionnement entre COAP et HTTP

La formation EFORT « M2M et Internet des Objets : Vision Réseau et Service » fournit toutes les clés de compréhension de l'écosystème M2M/IoT en terme de domaines d'application, architectures de réseau et de services associées, de protocoles utilisés et d'évolutions à venir.
http://www.efort.com/index.php?PageID=21&l=fr&f_id=169&imageField.x=5&imageField.y=7