Constrained Application Protocol

Asignatura:	ELO322
Fecha de entrega:	24-8-2018
Profesor:	Agustín González
Integrantes:	Bastian Cornejo
	Catalina Godoy
	Ernesto Roca

Resumen

En este trabajo abordamos el protocolo CoAP, el cual está pensado en el internet de las cosas y la limitación que estos objetos poseen. El proyecto lo llevamos a cabo con un servidor que envió mensajes CoAP a un respectivo cliente, y la captura posterior de estos mensajes.

Introducción

El Internet de las cosas (IoT, Internet of Things) es un concepto que se refiere a la interconexión digital de objetos cotidianos con Internet. Alternativamente, el Internet de las cosas es la conexión de Internet con más cosas u objetos que con personas. Si los objetos de la vida cotidiana tuvieran incorporadas etiquetas de radio, podrían ser identificados y gestionados por otros equipos, de la misma manera que si lo fuesen por seres humanos. El objetivo es hacer que todos estos dispositivos se comuniquen entre sí y, por consiguiente, sean más inteligentes e independientes. El Internet de las cosas es, además, significativo pues un objeto que se puede representar a sí mismo digitalmente se convierte en una mejora en comparación al objeto en sí; ya no se relaciona sólo con el usuario, sino que ahora se encuentra conectado a los objetos que lo rodean y a una base de datos.

Es completamente posible crear un sistema IoT a partir de las tecnologías Web existentes, a pesar de que no sean tan eficaces como lo sería con protocolos más nuevos. Dentro de estos, se pueden encontrar, por ejemplo, MQTT y algunas variantes como MQTT-S, XMPP, REST API y CoAP. Este último corresponde a un protocolo diseñado para objetos simples o con limitaciones a nivel de la capa de aplicación y posee una arquitectura basada en servicios web en lugar de una orientación a la mensajería.

IOT

Internet de las cosas trata la interconexión y transferencia de datos a través de internet de todo tipo de productor, sin requerir de interacciones externas. El objetivo es que distintos objetos de la vida cotidiana como un libro, un televisor, una puerta una ventana o un refrigerador, entre muchos otros, se comuniquen entre sí; aprovechando así el uso de internet. Esto con el fin de lograr control, monitoreo, comodidad y seguridad.

Actualmente, el Internet de las cosas es un tópico muy destacado dado que ofrece una gran cantidad de oportunidades y, al mismo tiempo, varios desafíos. La seguridad es el problema más común dentro de este tópico; junto con los millones de dispositivos que existen conectados a Internet y la gran cantidad de nuevos servicios que ofrecen, no sería raro dudar de la seguridad que estos poseen.

El internet de las cosas también permite el surgimiento de nuevas compañías de seguridad dedicadas a las distintas amenazas existentes. Aún así, sigue existiendo el problema de la privacidad y el intercambio de datos, tema bastante complicado cuando se habla de miles de millones de dispositivos conectados a Internet; ya que la mayoría de los dispositivos IoT están limitados, para abaratar sus costos y minimizar su tamaño, por lo que estos funcionan con las capacidades más simples de UDP y protocolos ligeros.

Protocolos de IoT

- **MQTT**: Actualmente de estándar abierto, es un protocolo de mensajería tipo publicación/suscripción muy ligero y pensado para dispositivos alimentados con baterías ya que tiene un reducido consumo de energía. Está diseñado para redes de comunicación poco fiables y es interesante comentar que posee tres modos de funcionamiento o “calidad de servicio” (QoS) .

    - QoS0 (At most once): El modo menos fiable pero también el más rápido. La publicación se envía pero no se recibe confirmación.

    - QoS1 (At least once): Se asegura que el mensaje es entregado al menos una vez, pero pueden recibirse duplicados.
    - QoS2 (Exactly once): El modo más fiable y que más ancho de banda consume. Se controlan los duplicados para garantizar que el mensaje es entregado una única vez.

  A pesar de sus características, MQTT puede suponer un problema para algunos dispositivos muy restrictivos, por el hecho de ir sobre TCP y de manejar nombres de topics largos. Esto se soluciona con la variante MQTT-SN que utiliza UDP y soporta indexación de nombres de topics.

- **REST**: HTTP y REST es muy útil para la mayoría de servicios y aplicaciones web, gateways y dispositivos. Es ampliamente utilizado y sencillo de implementar lo que facilita y acelera el desarrollo de muchos servicios en el internet de las cosas. Se puede utilizar de una forma muy simple, por ejemplo, para el intercambio de documentos JSON entre cualquier plataforma y tiene la ventaja de que un gran número de aplicaciones ya están desarrolladas utilizando esta tecnología.

Otro protocolo importante que se usa en IoT es CoAP, que se explicará a continuación.

CoAP

Es un protocolo software a nivel de aplicación pensado para ser usado en dispositivos electrónicos simples permitiendo que puedas comunicarse sobre Internet, está especializado en la transferencia web para usar nodos restringidos y del mismo modo, redes restringidas. Estos nodos usualmente tienen microcontroladores de 8-bit con una pequeña cantidad de ROM y RAM, mientras que las redes restringidas tal como IPv6 a través de Low-Power Wireless Personal Area Networks (6LoWPANs) a menudo tienen una gran tasa de paquetes erróneos y su rendimiento habitual es de 10x de kbit/s.

Está pensado para sensores de baja potencia y para aplicaciones destinadas a la intercambio machine to machine (M2M), como lo sería la energía inteligente y la automatización de construcciones, por otro lado se ha diseñado para trasladar el modelo HTTP, incluyendo otros requisitos como multicast, bajo overhead y simplicidad, que son muy importantes para el internet de las cosas (IoT) y Machine-to Machine (M2M).

Por otro lado este protocolo implementa el modelo REST de HTTP, el cual es un estilo de arquitectura software para sistemas hipermedia (conjunto de métodos para escribir, diseñar o componer contenidos) distribuidos como la World Wide Web (red informática mundial), por otro lado CoAP utiliza también cabeceras reducidas y limita el intercambio de mensajes, añadiendo soporte UDP.

Dicho protocolo también provee un modelo interacción de solicitud/respuesta entre la aplicación y los puntos finales, está diseñado con una interfaz HTTP simple para la integración web mientras cumple con los requisitos especializados de soporte de multicast, de muy bajo costo para la simplicidad de los entornos restringidos.

Características de CoAP

El grupo que diseñó el protocolo CoAP lo crearon con las siguientes características en mente:

- Sobrecarga y complejidad de análisis
- URI y soporte de tipos de contenido
- Soporte de descubrimiento de recursos provisto por servicios conocidos de CoAP
- Una simple suscripción de recursos y resultados de push notifications
- Almacenamiento simple en caché basado en la edad máxima

Formato de los mensajes

CoAP utiliza dos tipos de mensajes, petición y respuesta, utilizando un simple y binaria, formato base de encabezado. La base del encabezado puede seguir opciones de optimización de formato Type-Length-Value.

CoAP Message Structure

Byte								Byte								Byte								Byte							
0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
VER		TYPE		TKL (Token Length)				CoAP Request/Response Code								Message ID															
Token (If any in TKL bytes) (Maximum of 8 bytes)																															
Options (If Available) (Options Numbers: https://www.iana.org/assignments/core-parameters/core-parameters.xhtml#option-numbers)																															
1	1	1	1	1	1	1	1	Payload (If Available)																							

En la parte práctica implementamos un servidor en el IDE eclipse (ver Anexo, captura 1), el cual utilizaba lenguaje java, el cual transmitió mensajes CoAP al cliente (ver Anexo, captura 2), que implementamos con una extensión de Firefox llamada “Copper”. Ambos desde un mismo terminal.

Logramos que el servidor transmita al cliente un mensaje “Hello World” y capturamos los paquetes de la respectiva transmisión para analizarlos con Wireshark (ver Anexo, captura 3).

Como resultado comparamos lo conocido sobre los paquetes con los obtenidos, identificando sus partes.

Conclusión

CoAP puede ser un protocolo apropiado para dispositivos que operan con batería o mediante extracción de energía. CoAP aborda completamente las necesidades de un protocolo extremadamente ligero y con la naturaleza de una conexión permanente.

Tiene conocimiento semántico de HTTP y es RESTful (recursos, identificadores de recursos y manipula esos recursos a través de una Interfaz de programación de aplicaciones [API] uniforme). Además, si se cuenta con un entorno Web, usar CoAP será relativamente fácil.

Referencias

https://www.eclipse.org/californium/

https://tools.ietf.org/html/rfc7252

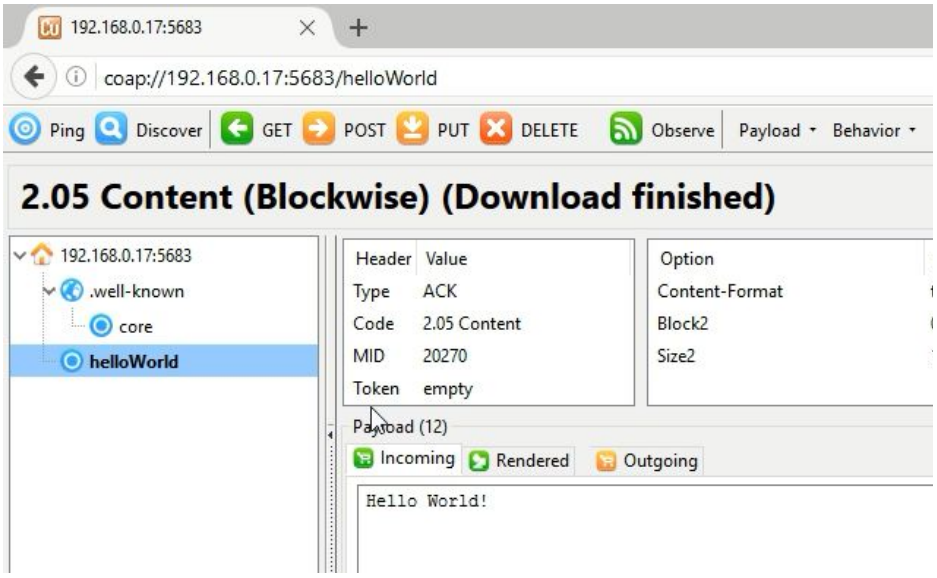
https://cs.iupui.edu/~xiaozhon/course_tutorials/Coap_tutorial_Arduino_Eclipse

https://www.cse.wustl.edu/~jain/cse574-14/ftp/coap/#sec3-1

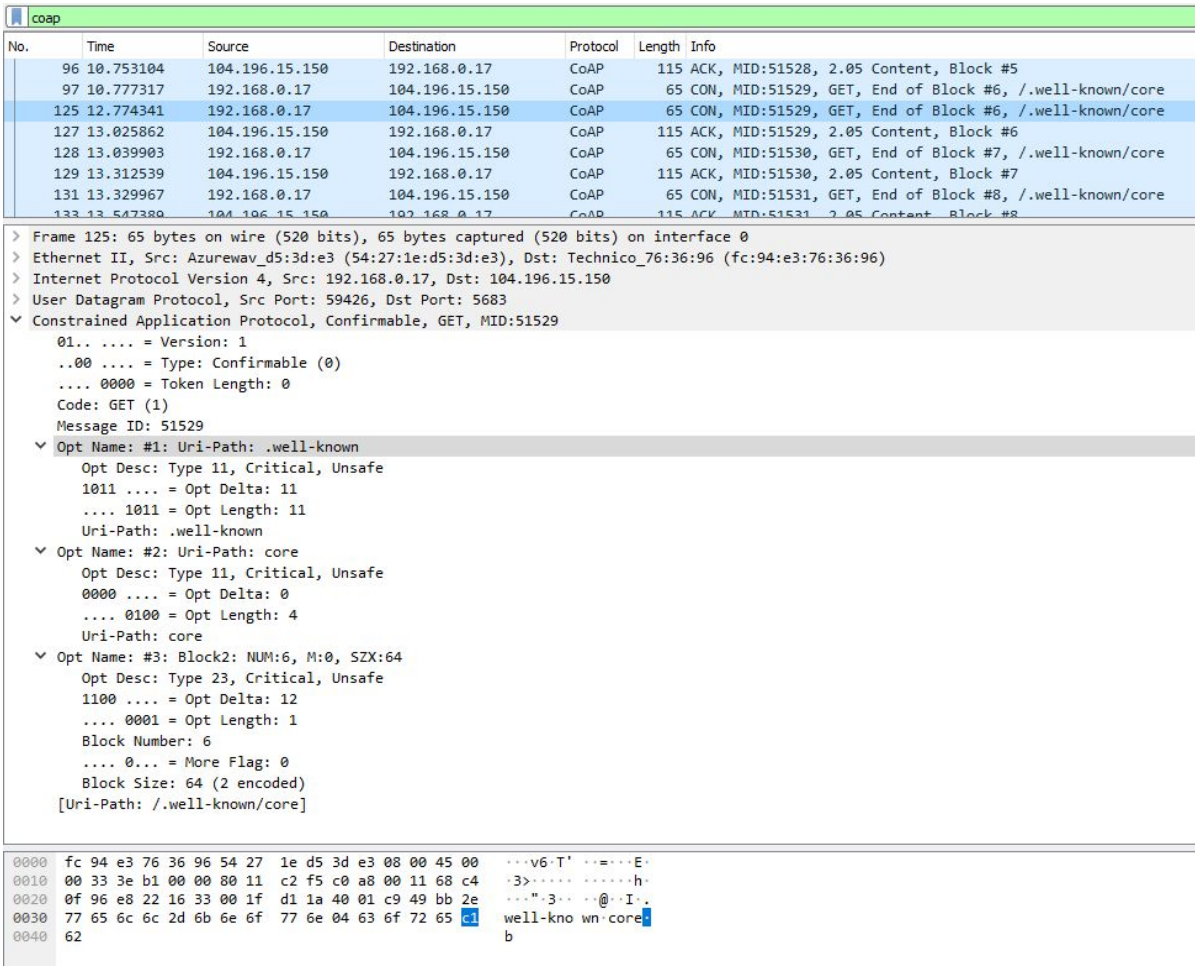
Anexo

```
2⊕ * Copyright (c) 2015 Institute for Pervasive Computing, ETH Zurich and others.
7  package org.eclipse.californium.examples;
8
9⊕ import java.net.Inet4Address;
0
1
2  public class HelloWorldServer extends CoapServer {
3
4      private static final int COAP_PORT = NetworkConfig.getStandard().getInt(NetworkConfig.Keys.COAP_PORT);
5⊖     /*
6       * Application entry point.
7       */
8⊖     public static void main(String[] args) {
9
0          try {
1
2              // create server
3              HelloWorldServer server = new HelloWorldServer();
4              // add endpoints on all IP addresses
5              server.addEndpoints();
6              server.start();
7
8          } catch (SocketException e) {
9              System.err.println("Failed to initialize server: " + e.getMessage());
0          }
1      }
2
3⊖     /**
4       * Add individual endpoints listening on default CoAP port on all IPv4 addresses of all network interfaces.
5       */
6⊖     private void addEndpoints() {
7          for (InetAddress addr : EndpointManager.getEndpointManager().getNetworkInterfaces()) {
8              // only binds to IPv4 addresses and localhost
```

Captura 1



Captura 2


coap
No. Time Source Destination Protocol Length Info
96 10.753104 104.196.15.150 192.168.0.17 CoAP 115 ACK, MID:51528, 2.05 Content, Block #5
97 10.777317 192.168.0.17 104.196.15.150 CoAP 65 CON, MID:51529, GET, End of Block #6, /.well-known/core
125 12.774341 192.168.0.17 104.196.15.150 CoAP 65 CON, MID:51529, GET, End of Block #6, /.well-known/core
127 13.025862 104.196.15.150 192.168.0.17 CoAP 115 ACK, MID:51529, 2.05 Content, Block #6
128 13.039903 192.168.0.17 104.196.15.150 CoAP 65 CON, MID:51530, GET, End of Block #7, /.well-known/core
129 13.312539 104.196.15.150 192.168.0.17 CoAP 115 ACK, MID:51530, 2.05 Content, Block #7
131 13.329967 192.168.0.17 104.196.15.150 CoAP 65 CON, MID:51531, GET, End of Block #8, /.well-known/core
Frame 125: 65 bytes on wire (520 bits), 65 bytes captured (520 bits) on interface 0
Ethernet II, Src: Azurewav_d5:3d:e3 (54:27:1e:d5:3d:e3), Dst: Technico_76:36:96 (fc:94:e3:76:36:96)
Internet Protocol Version 4, Src: 192.168.0.17, Dst: 104.196.15.150
User Datagram Protocol, Src Port: 59426, Dst Port: 5683
Constrained Application Protocol, Confirmable, GET, MID:51529
01.. .... = Version: 1
..00 .... = Type: Confirmable (0)
.... 0000 = Token Length: 0
Code: GET (1)
Message ID: 51529
Opt Name: #1: Uri-Path: .well-known
Opt Desc: Type 11, Critical, Unsafe
1011 .... = Opt Delta: 11
.... 1011 = Opt Length: 11
Uri-Path: .well-known
Opt Name: #2: Uri-Path: core
Opt Desc: Type 11, Critical, Unsafe
0000 .... = Opt Delta: 0
.... 0100 = Opt Length: 4
Uri-Path: core
Opt Name: #3: Block2: NUM:6, M:0, SZX:64
Opt Desc: Type 23, Critical, Unsafe
1100 .... = Opt Delta: 12
.... 0001 = Opt Length: 1
Block Number: 6
.... 0... = More Flag: 0
Block Size: 64 (2 encoded)
[Uri-Path: /.well-known/core]
0000 fc 94 e3 76 36 96 54 27 1e d5 3d e3 08 00 45 00 ···v6·T' ··=···E·
0010 00 33 3e b1 00 00 80 11 c2 f5 c0 a8 00 11 68 c4 ·3>····· ·····h·
0020 0f 96 e8 22 16 33 00 1f d1 1a 40 01 c9 49 bb 2e ···"·3·· ··@··I··
0030 77 65 6c 6c 2d 6b 6e 6f 77 6e 04 63 6f 72 65 c1 well-kno wn·core·
0040 62 b

Captura 3