Redes de sensores inalámbricos (RSI)

Capa de aplicación: CoAP

Leonardo Steinfeld

Inst. de Ingeniería Eléctrica, Fac. de Ingeniería
Universidad de la República (Uruguay)

NEON
Network of Competence on Internet of Things

Co-funded by the Erasmus+ Programme of the European Union

FACULTAD DE INGENIERÍA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Objetivos

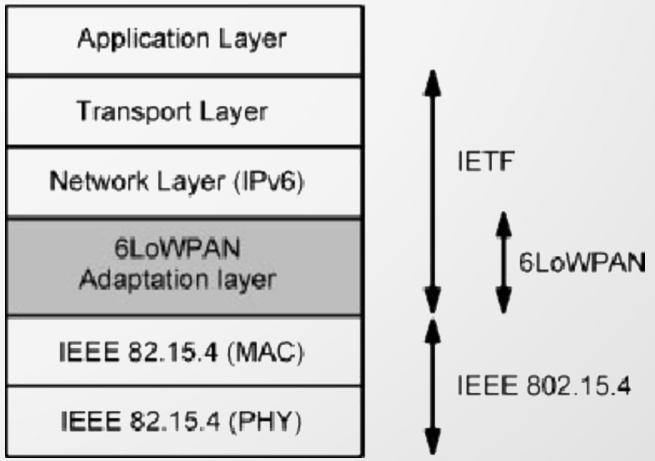
- Describir conceptos de la capa de aplicación en general
- Enumerar y describir las características de los principales protocolos
- Describir el formato y los mensajes de CoAP
- Describir las principales características de MQTT

Agenda

- Introducción y motivación
- Conceptos
  - REST (cliente-servidor)
  - Publish – Subscribe
- CoAP: mensajes, formato
- MQTT: generalidades
- Conclusiones

¿Dónde estamos?

- Top-down:
  - capa: **aplicación**
- Servicios disponibles (de capas inferiores)
  - TCP conexión a {dirección, puerto}
  - UDP datagrama a {dirección, puerto}
- Diferencias entre UDP y TCP
- ¿Qué podemos hacer?



Opciones

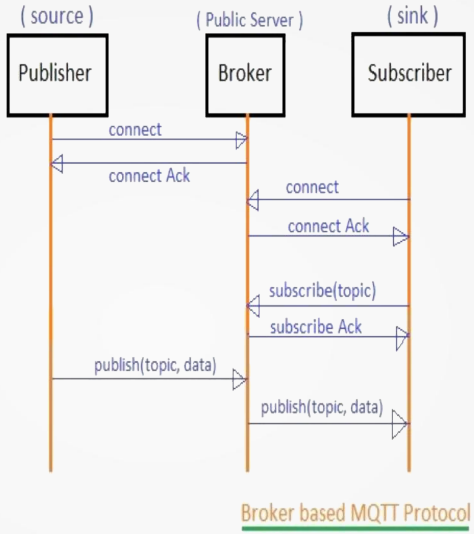
- Soluciones a medida o propietarias
  - Aproximación: pensar comunicación entre:
    - un nodo (device) y aplicación
    - dos sistemas embebidos

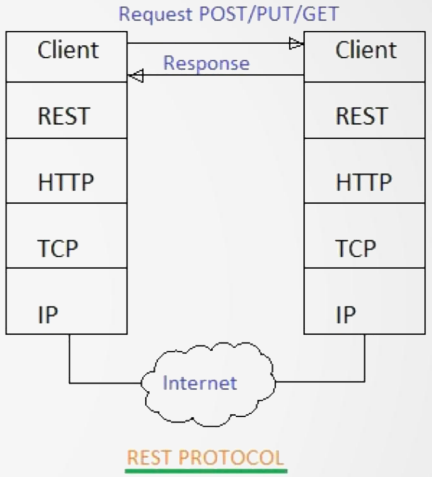
Propuesta de actividad grupal

- Diseñar un protocolo (bosquejo) de aplicación para:
  - obtener de nodos datos de sensores (más de uno) (por ejemplo: temperatura)
  - configurar el período de muestreo de los sensores.

Opciones

- Soluciones a medida o propietarias
  - desventajas
- Soluciones existentes
  - Cliente - Servidor
  - Publish – Subscribe





- Ejemplo: Extender el uso de web services (cliente-servidor)
  - Hypertext Transfer Protocol (HTTP)
  - “Nuevos” protocolos

Conceptos

- HTTP es un servicio web basado en modelo REST
- **REST (*RE*presentational *S*tate *T*ransfer)**
  - Transferencia de Estado Representacional
  - Arquitectura para sistemas aplicados para diseñar sistemas de **servicios web**
  - Arquitectura cliente/servidor

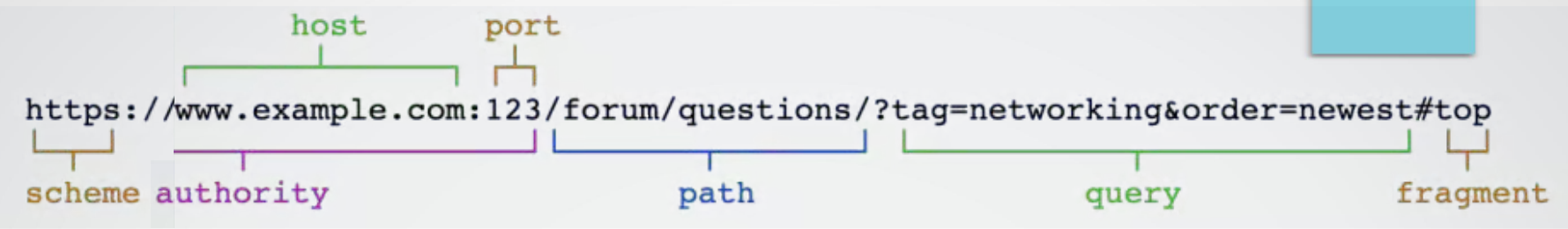
Conceptos

- **REST**
  - Servidores
    - no mantienen estados de transferencia (stateless)
    - no necesitan mantener diálogos abiertos
  - Cliente
    - Inicia consulta (petición):
      - Petición / Respuesta (Request/Response)

Conceptos

- Recurso:
  - abstracción controlada por un servidor
  - identificada por un *Universal Resource Identifier* (URI)
- Cada recurso
  - tiene una representación
    - contenido, valor o estado (en un momento dado)
- **REST** se basa en:
  - solicitudes y respuestas que transfieren representaciones de recursos.

Conceptos: uniform resource identifier



- Esquema (*scheme*):

  protocolo de acceso al recurso (e.g. http:, mailto:, coap:, etc.)

- Autoridad (*authority*):

  autoridad: dirección y puerto (e.g. www.home.com:80)

- Ruta (*path*):

  información de estructura jerárquica, que identifica al recurso (e.g. /casa/cocina)

- Consulta (*query*):

  información no jerárquica (pares "clave=valor"). Comienzo: '?'.

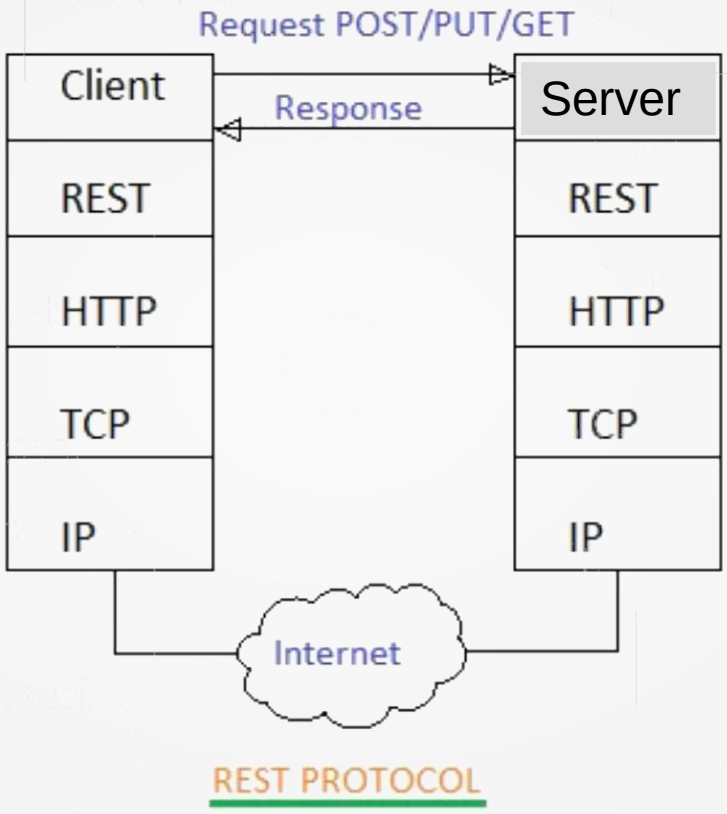
- Fragmento (*fragment*):

  para identificar una parte. Comienzo: '#'

Conceptos

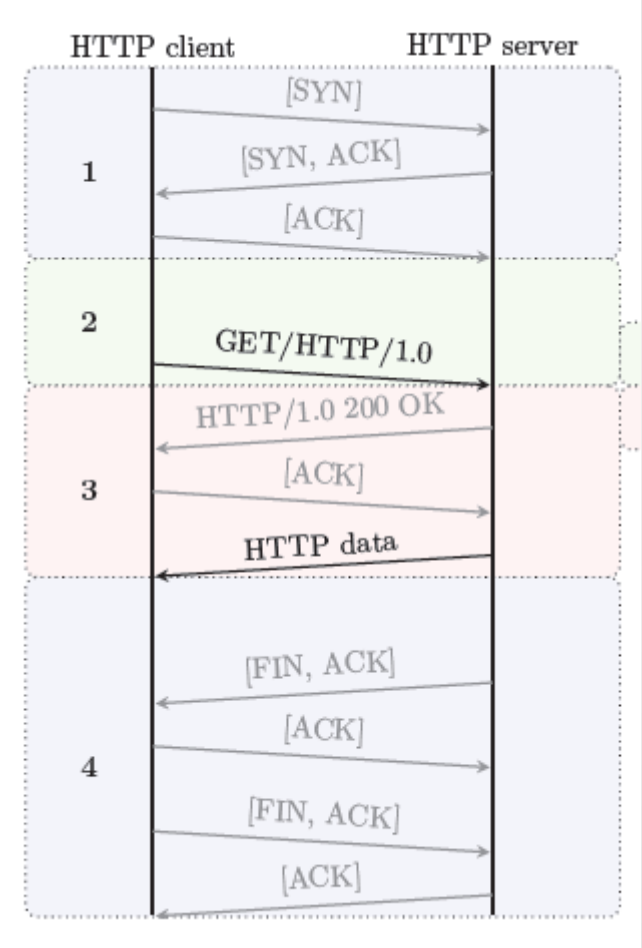
- Modelo REST
  - no especifica qué protocolo se debe utilizar, pero
  - requiere que un conjunto de métodos uniformes
- Métodos:
  - POST
  - GET
  - PUT
  - DELETE
- Correspondencia con: CRUD (Create, Read, Update, Delete)
  - funciones básicas en capa de persistencia (software)

Conceptos


Request POST/PUT/GET
Client
Response
Server
REST
HTTP
TCP
IP
REST
HTTP
TCP
IP
Internet
REST PROTOCOL

Ejemplo: HTTP

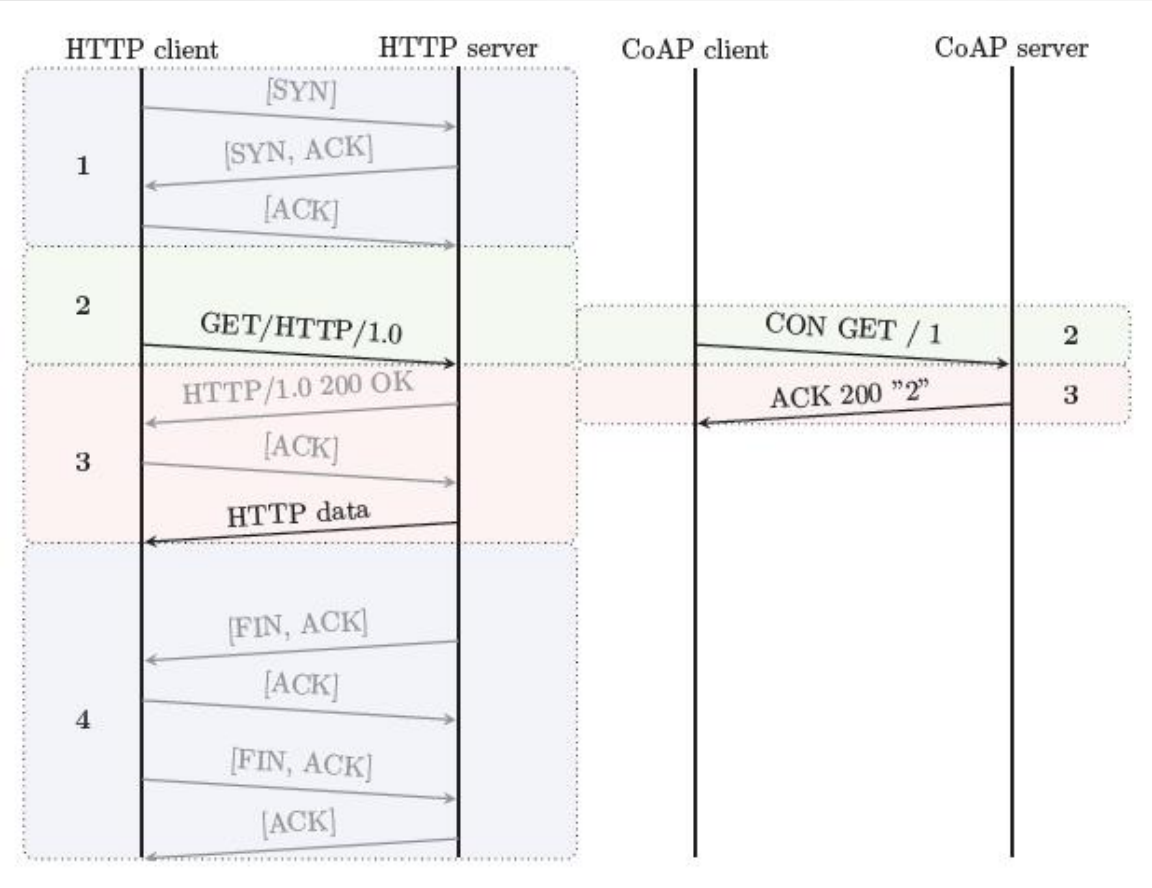
- Características
  - Transporte: TCP
- No apropiado para nodos pocos recursos
- Ejemplo:
  - Firefox → Developer → Network (Ctrl+Shift+E)
  - visitar:
    - www.fing.edu.uy y buscar
    - iie.fing.edu.uy/~leo/rsi.html



Nueva propuesta: CoAP

- IETF WG Constrained RESTful Environments (core)
  - RFC 7252: The **Co**nstrained **A**pplication **P**rotocol (CoAP)
    - RESTful pero diseñado desde cero
- Transporte: UDP (opcionalmente DTLS)

CoAP vs HTTP


HTTP client
HTTP server
CoAP client
CoAP server
[SYN]
[SYN, ACK]
[ACK]
1
2
GET/HTTP/1.0
CON GET / 1
2
HTTP/1.0 200 OK
ACK 200 "2"
3
[ACK]
3
HTTP data
[FIN, ACK]
[ACK]
4
[FIN, ACK]
[ACK]

Ejemplo de URIs

- coap:// host [:port] /path ["?" query]
- coap://[aaaa::212:4b00:430:501d]:**5683**/node/sensors/air_temperature?type=measure

  coap://[aaaa::212:4b00:430:501d]:5683/node/sensors/air_temperature?type=measure
- coap://[aaaa::212:4b00:430:501d]:5683/node/sensors/air_temperature?type=sample_period

  coap://[aaaa::212:4b00:430:501d]:5683/node/sensors/air_temperature?type=sample_period

Diseño CoAP

- Repaso: modelo cliente/servidor
  - M2M (machine-to-machine): típico ambos roles:
  - cliente:
    - envía **request** de una acción sobre un recurso usando un **método**.
  - servidor:
    - envía **response**
      (puede incluir representación del recurso).

Diseño: dos capas

- Métodos
  - interacción request/response
- Mensajes:
  - interacciones asíncronas
  - UDP como transporte

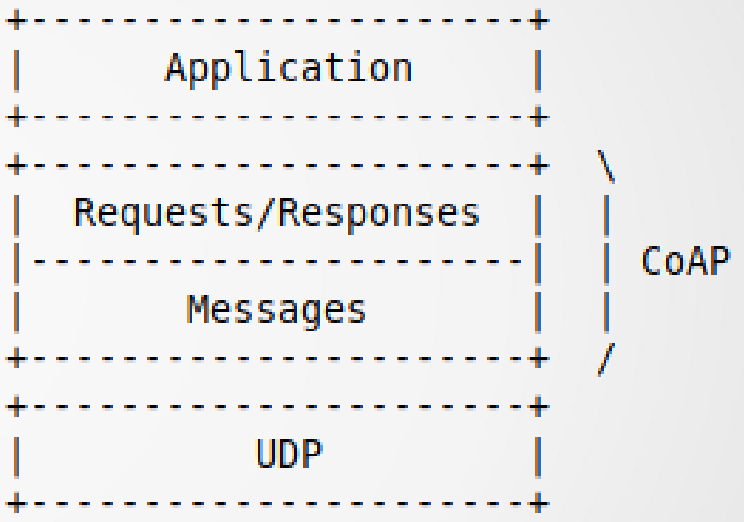


Figure 1: Abstract Layering of CoAP

Diseño: dos capas

- Mensajes (tipos)
  - Confirmable
  - Non-confirmable
  - Acknowledgement
  - Reset
- Métodos
  - GET
  - PUT
  - POST
  - DELETE

Mensajes: Confirmable (CON)

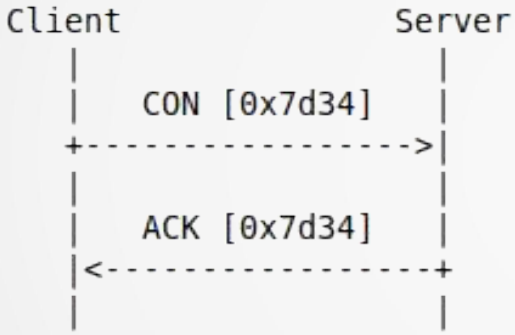


Figure 2: Reliable Message Transmission

- Confirmable (confiable)
  - se retransmite hasta tener respuesta
    - ACK
    - RESET
  - *Message ID*
    - identificar resp.

Mensajes: No-Confirmable (NON)

- No-Confirmable

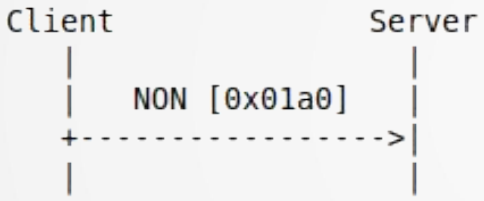


Figure 3: Unreliable Message Transmission

Modelo request / response: inmediata

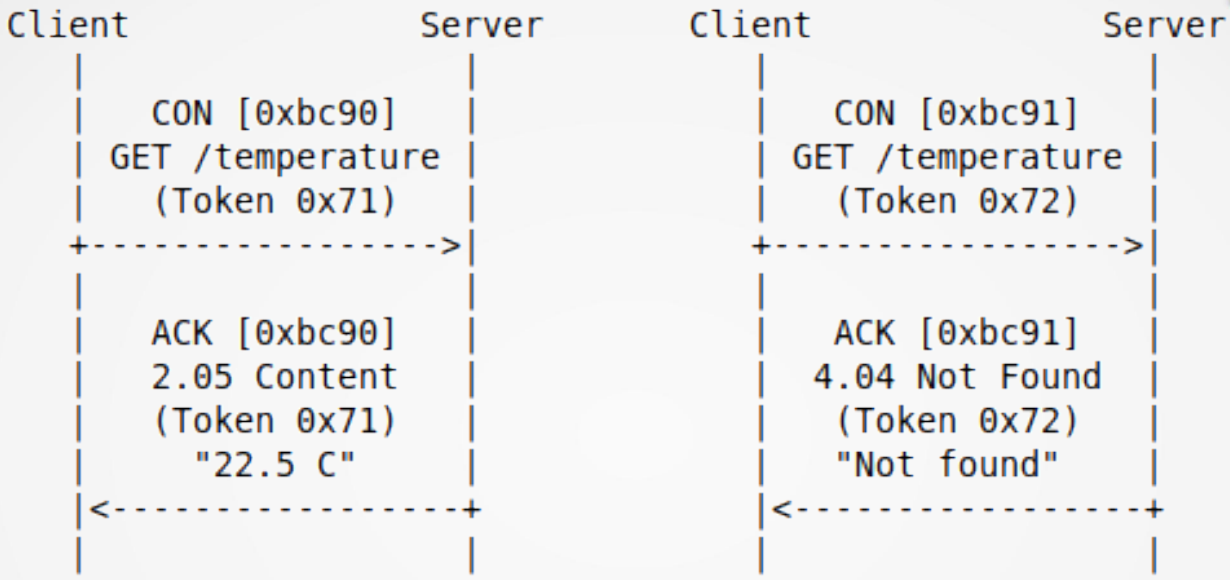


Figure 4: Two GET Requests with Piggybacked Responses

- *Token*
  - asocia response a request. (indep. de mensajes)

Modelo request / response: separada

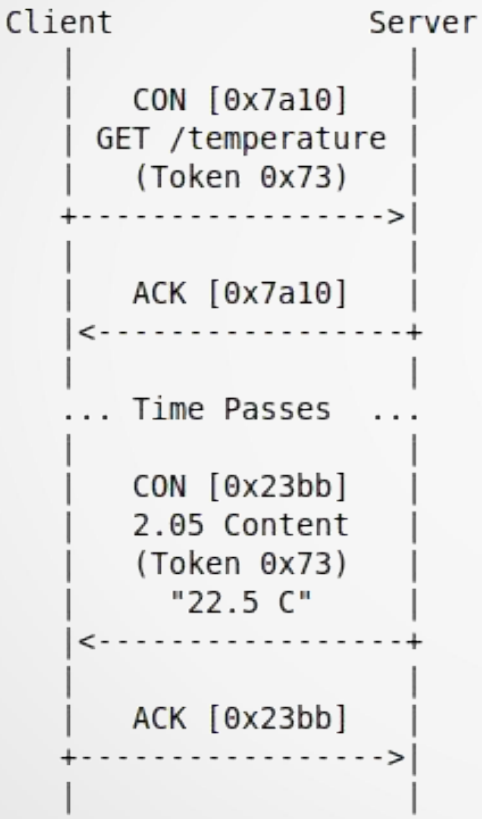


Figure 5: A GET Request with a Separate Response

- Respuesta separada
  - servidor aun no tiene respuesta
  - mensaje confirmable
- Observar:
  - *Token*
  - *Message ID*

Modelo request / response: observable

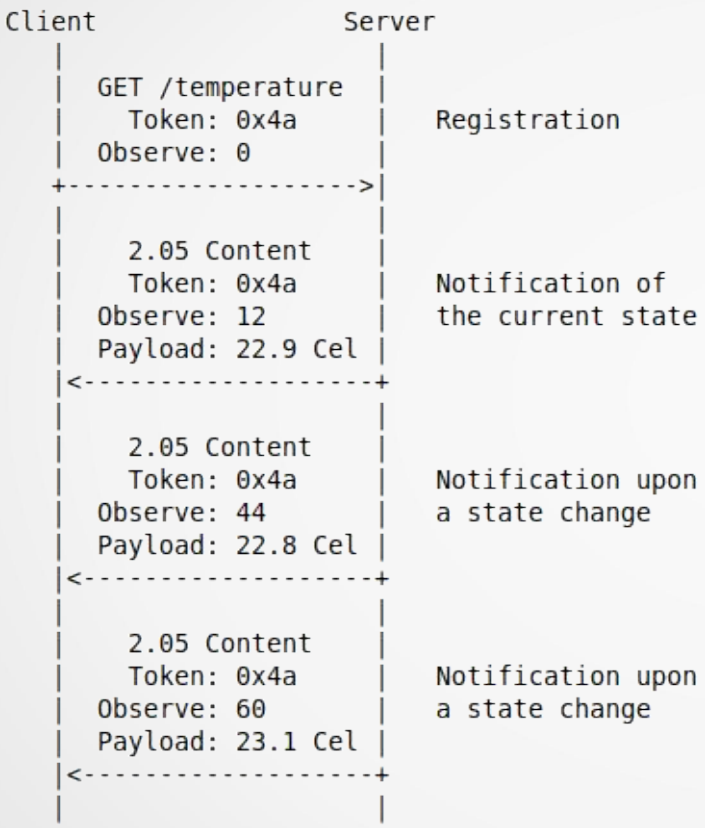


Figure 2: Observing a Resource in CoAP

- Observable (RFC 7641):
  - patrón: publish-subscribe
- Observar:
  - *Token*
  - *Observe*

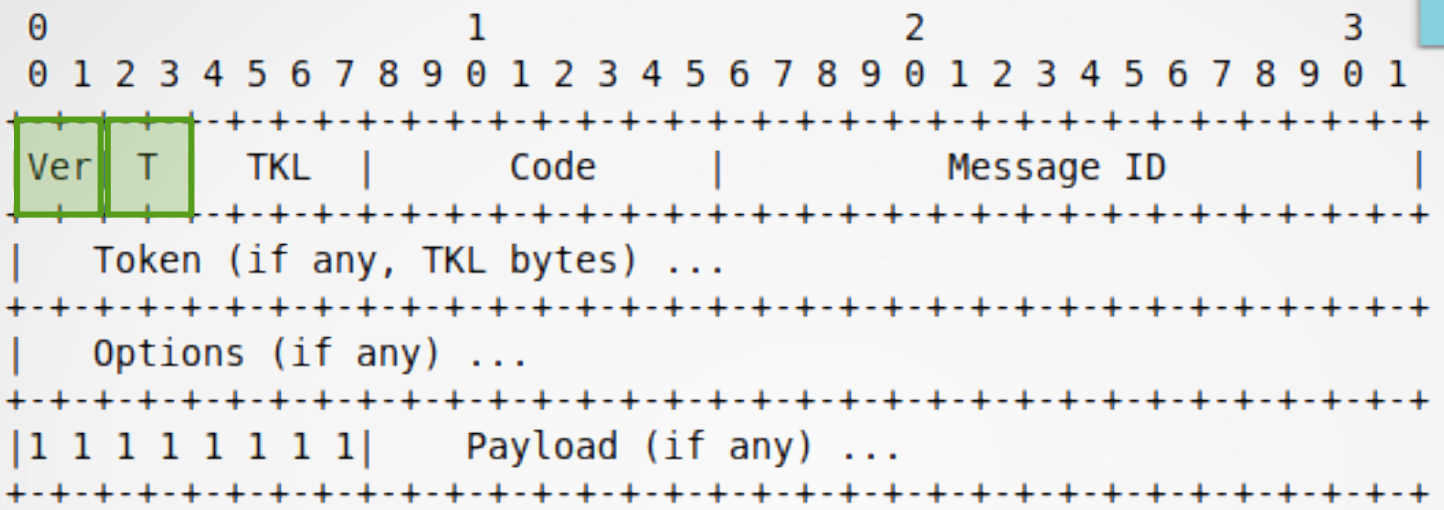
RFC 7641 “Observing Resources in the Constrained Application Protocol (CoAP)”

Diseño: resumen

GET, **POST**, **PUT**, **DELETE**
URIs and Internet Media Types

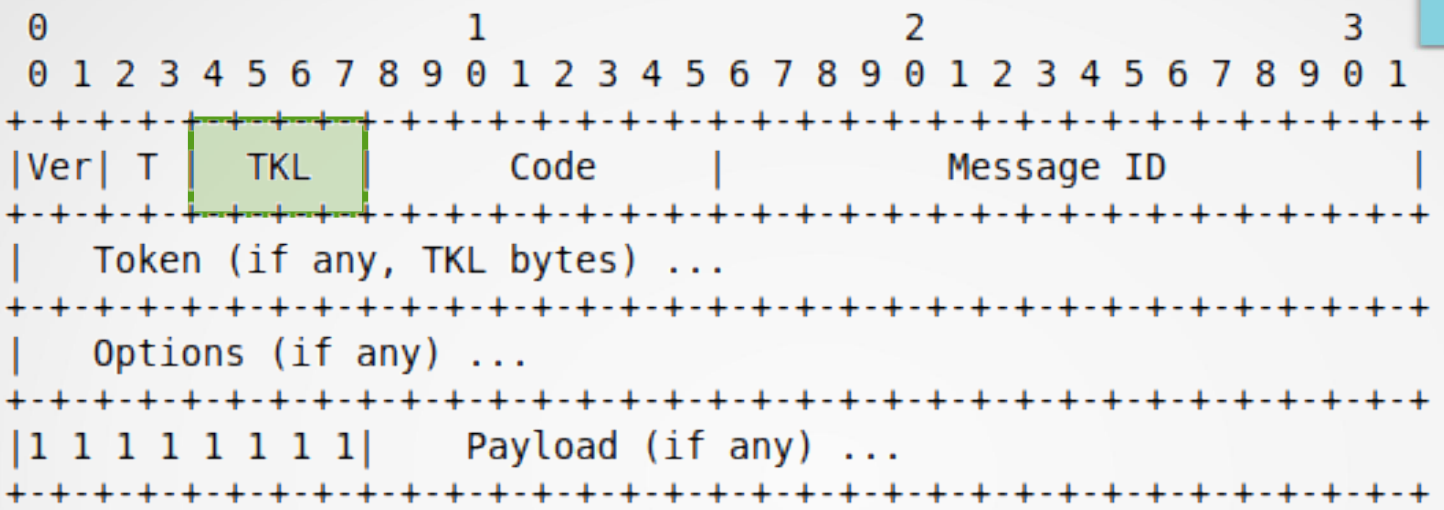
Deduplication
Optional retransmissions
(Confirmables “CON”)

CoAP: formato



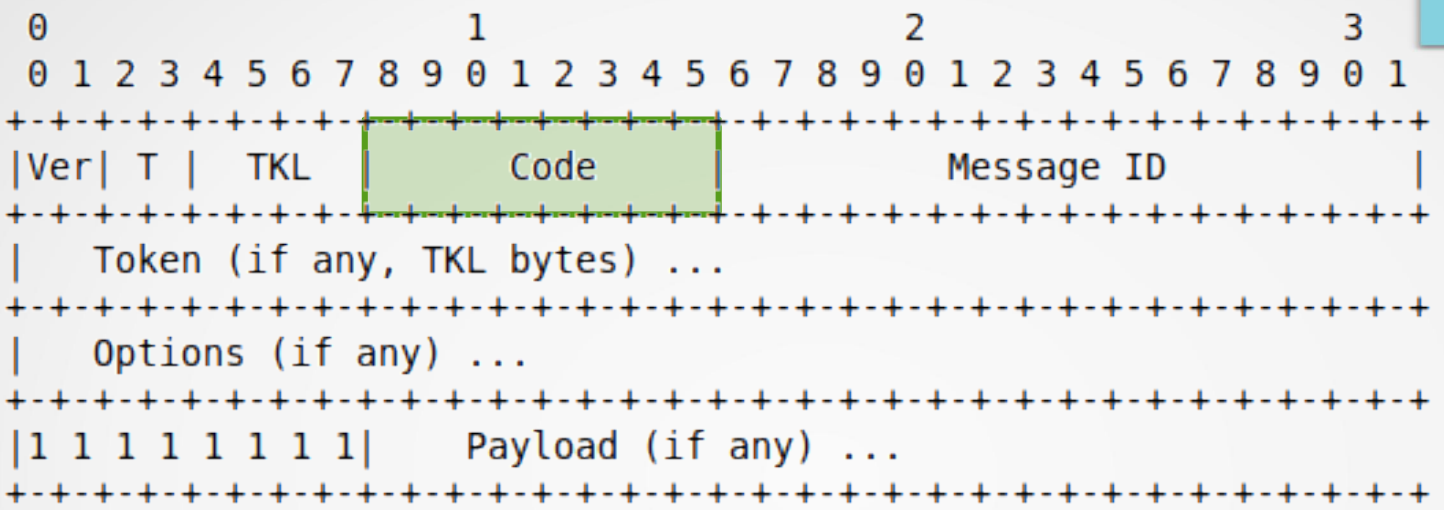
- *Version (Ver):* (2 bits)
  - versión actual 1 (binario 01)
- *Type (T): Tipo de mensaje* (2 bits)
    - Confirmable (0), Non-confirmable (1), Ack (2), Reset (3)

CoAP: formato



- *Token Length (TKL):* (4-bit)
  - Inidca el largo del campo Token de largo variable
- Nota:
  - El largo del payload se calcula a partir del tamaño del datagrama

CoAP: formato



- *Code: Dividido en "c.dd"*
  - *class* (3-bit)*:*
    - *request (0), a success response (2), a client error response (4), or a server error response (5)*
  - *detail* (5-bit)

Request / Response codes

- *Code: "c.dd" (class . detail)*

request

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

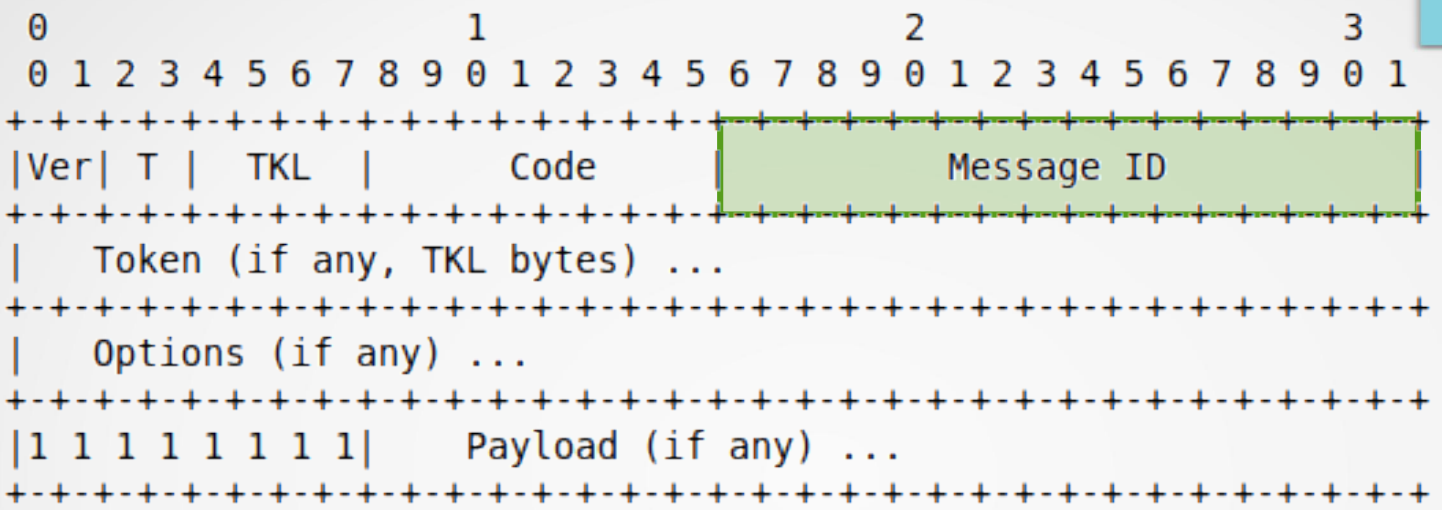
Table 5: CoAP Method Codes

response

	Code	Description	Reference
success	2.01	Created	[RFC7252]
	2.02	Deleted	[RFC7252]
	2.03	Valid	[RFC7252]
	2.04	Changed	[RFC7252]
	2.05	Content	[RFC7252]
client error	4.00	Bad Request	[RFC7252]
	4.01	Unauthorized	[RFC7252]
	4.02	Bad Option	[RFC7252]
	4.03	Forbidden	[RFC7252]
	4.04	Not Found	[RFC7252]
	4.05	Method Not Allowed	[RFC7252]
	4.06	Not Acceptable	[RFC7252]
	4.12	Precondition Failed	[RFC7252]
	4.13	Request Entity Too Large	[RFC7252]
	4.15	Unsupported Content-Format	[RFC7252]
server error	5.00	Internal Server Error	[RFC7252]
	5.01	Not Implemented	[RFC7252]
	5.02	Bad Gateway	[RFC7252]
	5.03	Service Unavailable	[RFC7252]
	5.04	Gateway Timeout	[RFC7252]
	5.05	Proxying Not Supported	[RFC7252]

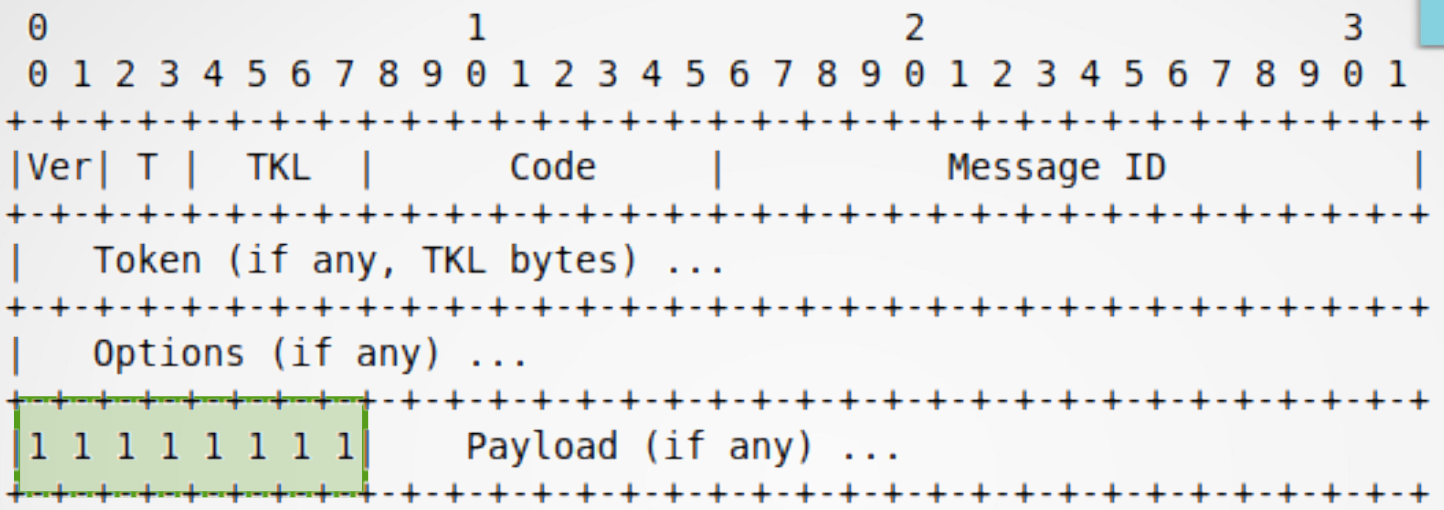
Table 6: CoAP Response Codes

CoAP: formato



- *Message ID:*
  - para detectar duplicados y asociar Ack/Reset a mensajes Confirmable/Non-confirmable.

CoAP: formato



- *Payload Marker (0xFF)*
  - Indica el fin de las opciones y el comienzo del payload

Payload: *media type*

Media type	Encoding	ID	Reference
text/plain; charset=utf-8	-	0	[RFC2046] [RFC3676] [RFC5147]
application/link-format	-	40	[RFC6690]
application/xml	-	41	[RFC3023]
application/octet-stream	-	42	[RFC2045] [RFC2046]
application/exi	-	47	[REC-exi-20140211]
application/json	-	50	[RFC7159]

Table 9: CoAP Content-Formats

- Especificado como opción

Otras consideraciones

- Descubrimiento de recursos
  - path: /.well-known/core

- Transferencia de bloques (RFC 7929)
  - blockwise transfer

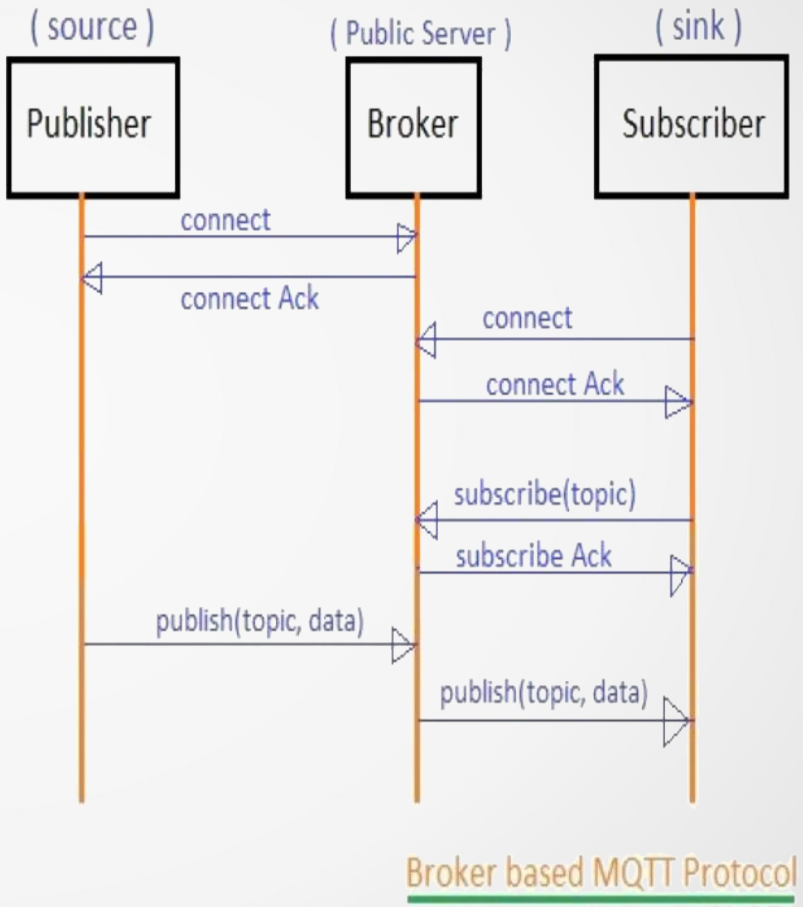
RFC 7929 “Block-Wise Transfers in the Constrained Application Protocol (CoAP)”

Normalización

- IETF WG Constrained RESTful Environments (core)
  - RFC 7252: “The Constrained Application Protocol (CoAP)”
  - RFC 7641: “Observing Resources in the Constrained Application Protocol (CoAP)”
  - RFC 7929: “Block-Wise Transfers in the Constrained Application Protocol (CoAP)”

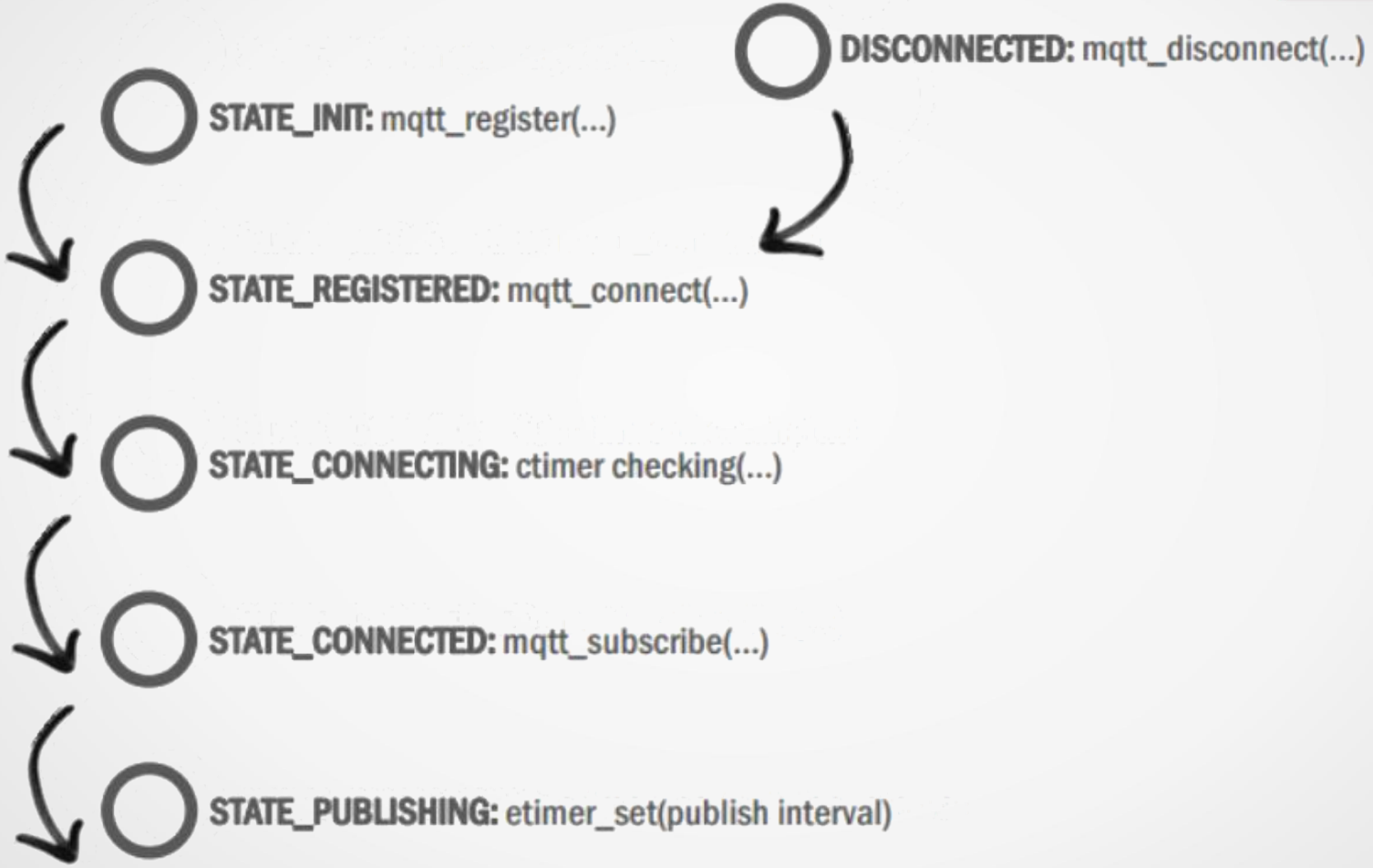
MQTT

- Publish Subscribe
  - desacopla:
    - envío
    - recepción
- Open standard
- Capa de transporte: TCP

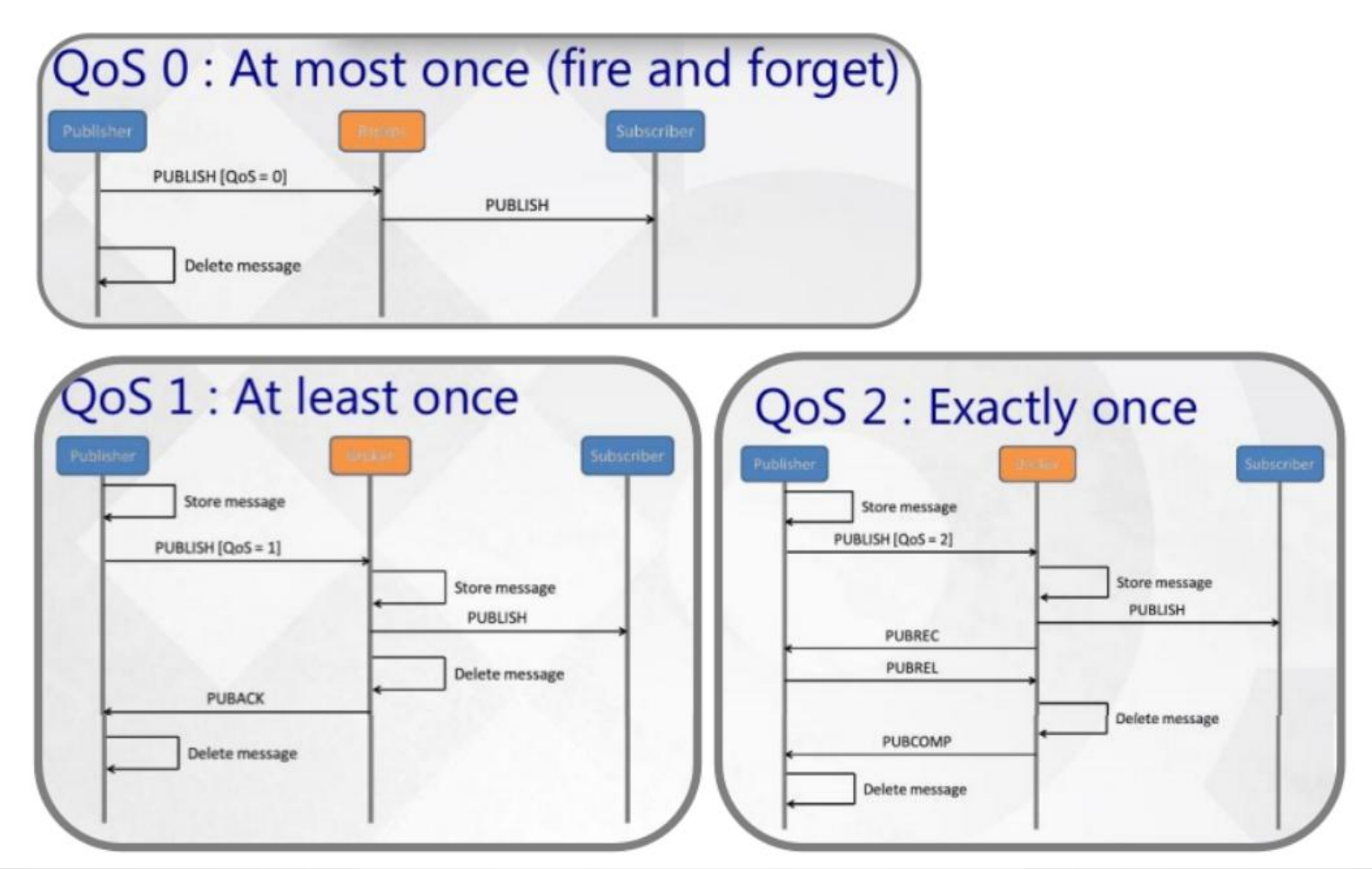


Broker based MQTT Protocol

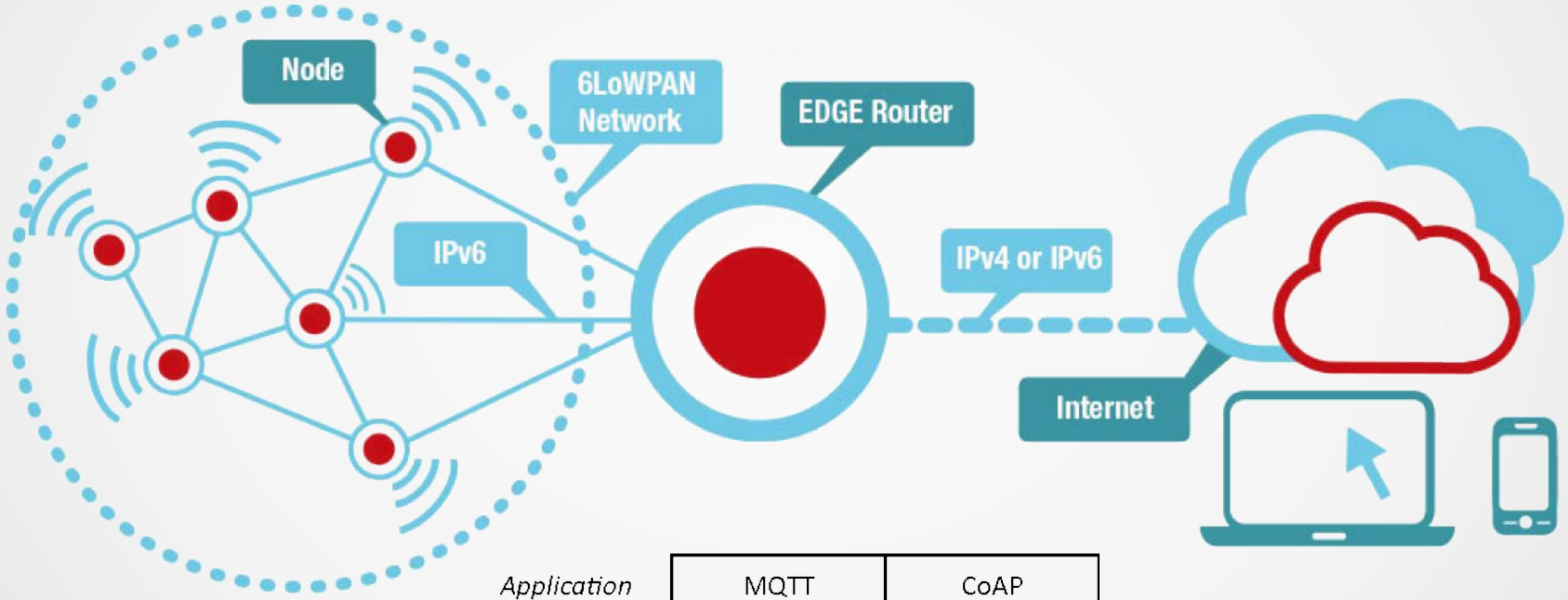
MQTT: estados (Contiki)



MQTT: QoS


QoS 0 : At most once (fire and forget)
Publisher
Subscriber
PUBLISH [QoS = 0]
PUBLISH
Delete message
QoS 1 : At least once
Publisher
Subscriber
Store message
PUBLISH [QoS = 1]
Store message
PUBLISH
Delete message
PUBACK
Delete message
QoS 2 : Exactly once
Publisher
Subscriber
Store message
PUBLISH [QoS = 2]
Store message
PUBLISH
PUBREC
PUBREL
Delete message
PUBCOMP
Delete message

MQTT en 6lowpan



Application	MQTT	CoAP
Transport	TCP	UDP
Network	IPv6	
Adaptation	6LoWPAN	
Link Physical	IEEE 802.15.4	Bluetooth Low Energy

Bibliografía

- RFC 7252 y documentos asociados
- M. Kovatsch, S. Duquennoy and A. Dunkels, "A Low-Power CoAP for Contiki," 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, Valencia, 2011, pp. 855-860.
- Johan Westö & Dag Björklund, *An Overview of Enabling Technologies for THE INTERNET OF THINGS*, Novia University of Applied Sciences, Novia publikation och produktion, serie R: Rapporter 1/2014

Planificación clases

1) Introducción RSI
2) Plataformas de hardware
3) Arquitectura 6LoWPAN (IPv6)
4) Plataforma de software: Contiki-NG (parte 1)
5) Plataforma de software: Contiki-NG (parte 2)
6) **Capa de aplicación: CoAP / MQTT**
7) Capa de red: RPL
8) MAC
9) IEEE 802.15.4 / 6lowpan
10) Capa Fisica & antenas
11) IoT y las RSI

FIN... ¿más preguntas?