ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

INGENIERO ELECTROMECÁNICO

SNMP VS COAP EN SISTEMAS DE DISTRIBUCIÓN ELÉCTRICA

Autor: Pablo Maceda Dal-re
Director: José Antonio Rodríguez Mondéjar

Madrid
Julio 2016

Índice de la memoria

SNMP vs CoAP en sistemas de distribución eléctrica

Índice de figuras

Índice de tablas

Parte I MEMORIA

Capítulo 1 INTRODUCCIÓN

1.1 ESTADO DEL ARTE

1.1.1 SNMP

SNMP es un protocolo clásico que apareció en la década de 1980 perteneciente a los denominados protocolos de internet (comúnmente llamados TCP/IP). Está orientado a la monitorización y administración de los dispositivos de una red como routers, impresoras, switches, servidores, etc. Se trata de un protocolo de la capa de aplicación y que usa UDP, TCP e IP.

Al ser un protocolo que lleva tanto tiempo instaurado y regulado ha tenido varias versiones, en concreto 3 principales (SNMPv1, SNMPv2 y SNMPv3):

-La primera versión del protocolo comenzó a ser regulada con los primeros RFCs en 1988. Una de las mayores deficiencias de esta primera versión fue la falta de seguridad, ya que en aquella época fue aprobado basándose en la consigna de que SNMP fuera un protocolo provisional necesario para la expansión a gran escala de Internet.

-La segunda versión (SNMPv2) surgió a mediados de los años 90 y supuso una mejora en rendimiento, confidencialidad, seguridad y comunicaciones manager-to-manager. Introdujo dos nuevos tipos de PDU (GetBulkReques e InformRequest) que ayudaban a la fiabilidad y rendimiento del protocolo. Las mejoras iniciales en seguridad eran demasiado complejas e impidieron la expansión de SNPv2 hasta la creación del SNMPv2c que mantenía el resto de mejoras, pero usaba el mismo sistema de seguridad de SNMPv1.

-La tercera, y hasta ahora más reciente, versión (SNMPv3) dio un gran salto en la seguridad sin modificar realmente mucho el protocolo ya existente en su versión anterior. Añadió también mejoras a la configuración remota de los dispositivos de la red. Desde 2004 la IETF reconoce a SNMPv3 como la

versión actual y le otorga un "Internet Standard", considerando las versiones anteriores como obsoletas.

1.1.2 CoAP

CoAP es un protocolo también de la capa de aplicación, que permite la comunicación de aparatos electrónicos a través de internet. Es un protocolo inspirado en y fácilmente traducible al HTTP, para una fácil integración en la web. Además, cumple con ciertos requisitos como el soporte multidifusión, simplicidad y un bajo consumo todos ellos muy importantes para el Internet of Things y las comunicaciones M2M. CoAP puede implementarse en casi todos los aparatos que usen UDP o un análogo.

CoAP es un protocolo muy reciente, ni siquiera ha recibido aún un Internet Standard de la IETF. El RFC7252, una "propuesta de estándar", publicado en junio de 2014 es donde se recoge la mayoría de las indicaciones sobre el protocolo.

1.1 IoT

1.1.3 Otros protocolos no estudiados en el trabajo

1.1.3.1 OMA LWM2M

OMA LWM2M (Open Mobile Alliance Lightweight Machine to Machine) es un protocolo de la capa de aplicación diseñado por la Open Mobile Alliance, para el manejo de aparatos M2M o Internet of Things.

1.1.3.2 CMIP

CMIP (Common Management Information Protocol) diseñado a la par y como competidor del SNMP. Más complejo y tiene más funciones que el SNMP, pretendía ser la solución a largo plazo de éste. En la actualidad es utilizado en telecomunicaciones y como principal protocolo de comunicación para sistemas que utilizan el modelo OSI.

1.2 MOTIVACIÓN DEL PROYECTO

El SNMP es un protocolo clásico que lleva implantado desde los años 80 y que, a pesar de estar muy extendido en las redes de comunicación, tiene carencias debido a su falta de actualizaciones y mejoras desde su última versión en 2004. El Internet of Things es probablemente el futuro de las comunicaciones y CoAP podría ser el protocolo que acabe sustituyendo a SNMP para la organización de redes de comunicación. Para ello este trabajo pretende hacer una comparación entre ambos para poder determinar si CoAP podría llegar a tomar el relevo de SNMP en el marco de las redes de comunicación ubicadas, por ejemplo, en las redes de distribución eléctrica.

1.3 OBJETIVOS

El objetivo claro de este proyecto es a comparación de ambos protocolos, para ello habrá varios objetivos intermedios:

- Primero un estudio profundo de SNMP, compresión de su funcionamiento, detalle, normativa e implementación.
- Puesta en práctica en el laboratorio de SNMP en componentes similares a los encontrados en los sistemas industriales que forman una red de distribución eléctrica.
- Estudio en profundidad del protocolo CoAP, comprensión de su funcionamiento, normativa e implementación.
- Implementación en el laboratorio del protocolo CoAP con la tecnología que podamos encontrar, prueba de su funcionamiento.
- Comparativa entre ambos protocolos, partes en común de ambos, ventajas e inconvenientes que tiene uno sobre el otro.

1.4 METODOLOGÍA / SOLUCIÓN DESARROLLADA

Mucha parte del proyecto es de estudio. Investigación online, en la base de datos de la IETF, lectura de los RFCs, libros sobre el tema para recopilar información, estudiar y entender las características, normas y el funcionamiento de ambos protocolos.

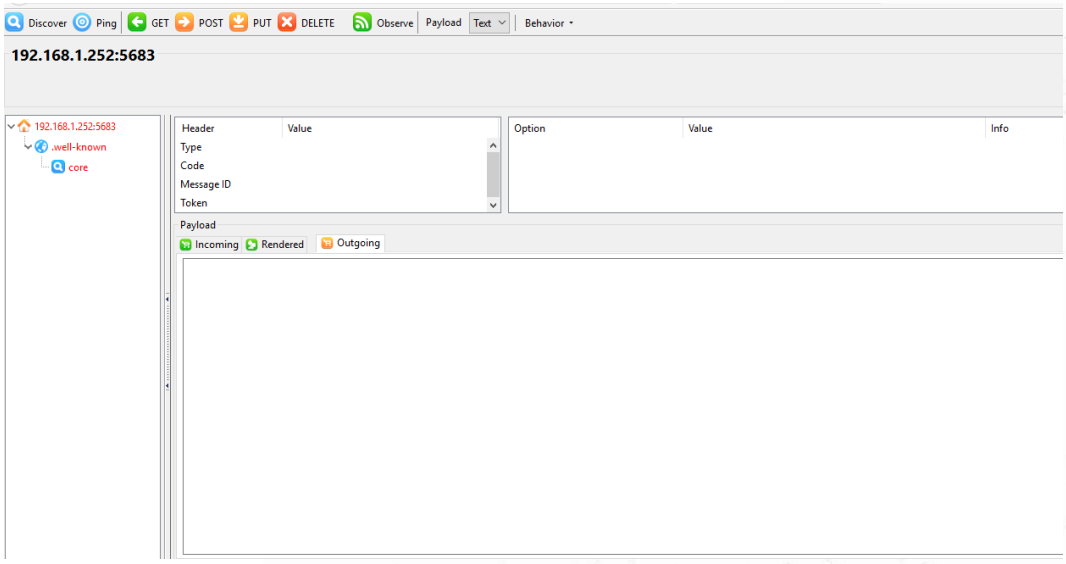
La parte práctica del SNMP se desarrollará en el Laboratorio de Automatización donde se dispone de los componentes y programas necesarios para su implementación. La parte práctica de CoAP se desarrollará desde un ordenador donde se hayan podido instalar las herramientas necesarias.

1.5 RECURSOS / HERRAMIENTAS EMPLEADAS

Para la parte de SNMP se ha utilizado la herramienta TIA Portal V13 disponible en el laboratorio de automatización. El programa de serie no permite la implantación de este protocolo, por lo que para poder operar con él hizo falta la descarga e implementación de una librería especifica proporcionada por Siemens (los desarrolladores de TIA Portal) que se puede encontrar en el siguiente link:

Link librería SNMP TIA Portal

Para la parte de CoAP se ha utilizado la herramienta Copper (Cu), un *Add-On* para el navegador Mozilla Firefox desarrollado como un proyecto por Matthias Kovatsch de la universidad ETH Zurich. Este *Add-On* permite el control de cualquier elemento a través CoAP. Simplemente introduciendo una URI de un servidor CoAP (coap://…) convierte el navegador en una interfaz para la interacción con dicho servidor.



1.2 Cooper para Firefox

Link de descarga de Cooper

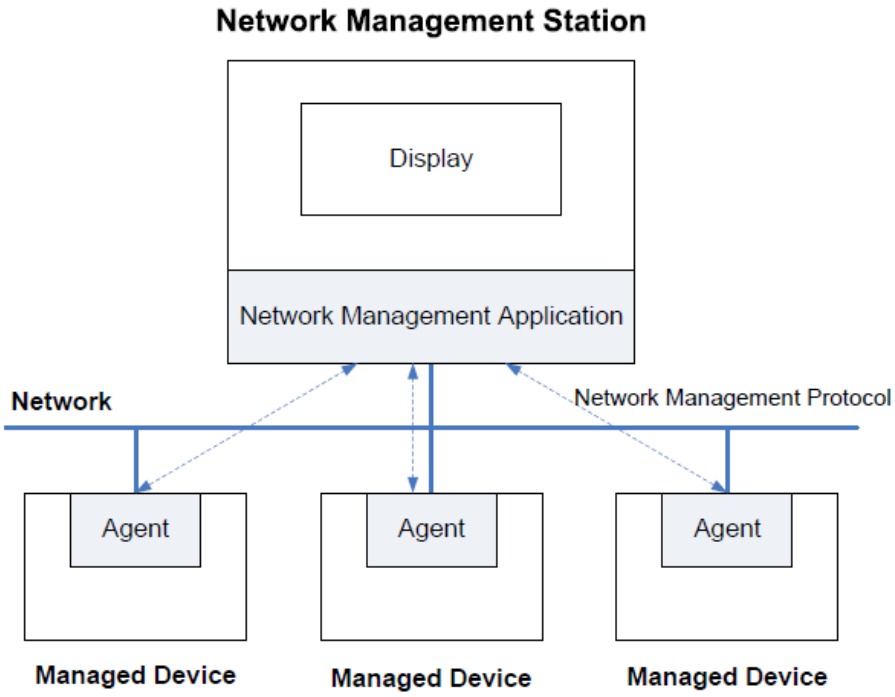
Capítulo 2 SNMP

2.1 INTRODUCCIÓN

Network Management es un servicio que emplea una variedad de protocolos, herramientas y aplicaciones para asistir a las personas encargadas de la administración, monitorización y control del correcto funcionamiento de los recursos de una red, tanto hardware como software, para proporcionar los servicios y cumplir los objetivos requeridos a la red.

Cuando se desarrolló TCP/IP no se prestó casi ninguna atención al *Network Management*. Hasta la década de los 80 la gestión de las redes era individual y manual, no se regía por ningún protocolo ni convenio acordado y programarlo con las tecnologías de la época suponía un gran esfuerzo innecesario. A partir de los 80, debido al aumento del tamaño y complejidad de las redes, se empezaron a extender y difundir las diferentes tecnologías desarrolladas para la administración de redes. El punto de partida fue en noviembre de 1987 cuando se publicó el SGMP que sería la base sobre la que, un año más tarde, la IAB aprobó la primera versión del *Simple Network Management Protocol* (SNMP) como una solución a corto plazo para el *Network Management*. Estándares como el SNMP y el CMIP allanaron el camino para aplicaciones y herramientas estandarizadas para la administración de redes de comunicación.

Un Network Management system (NMS) consiste en una serie de aplicaciones que permiten monitorizar y controlar los componentes de una red de comunicación. En general, casi todos los NMS tienen la misma arquitectura como la que se muestra en la figura 2.1.



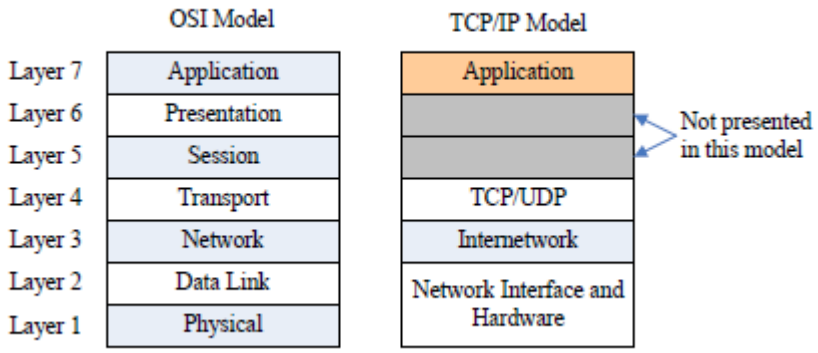
2.1 Arquitectura de "Network Management Systems"

Esta arquitectura tiene dos principales componentes: el dispositivo controlador, llamado *Network Management Station* o Manager a secas y el dispositivo controlado llamado Agente. El Manager sirve como interfaz entre la persona y el NMS a controlar. También es la plataforma para las diferentes aplicaciones diseñadas para efectuar funciones de control y monitorización de los Agentes.

Dada la diversidad de elementos a controlar, tales como hubs, routers, puentes de red, switches…etc., y la gran variedad de sistemas operativos e interfaces de programación disponibles, un protocolo es imprescindible para que el Manager sea capaz de comunicarse con los Agentes de manera efectiva. SNMP y CMIP son dos protocolos extendidos de este estilo, aunque debido a su mayor expansión en este trabajo nos centraremos en el SNMP.

Modelo OSI vs Modelo TCP/IP

Existen dos modelos de referencia para los protocolos de red de arquitectura en capas: el modelo OSI creado por la ISO y el TCP/IP, la estructura de ambos modelos puede ser observada en la figura 2.2.



2.2 Modelos de referencia OSI y TCP/IP

El modelo OSI fue desarrollado con la premisa que las diferentes capas de protocolos tuvieran diferentes funciones. El modelo OSI dispone de 7 capas mientras que el TCP/IP reduce el número a 4. Ambos protocolos tienen mucho en común, ambos se basan en el concepto de agrupar protocolos independientes y las funciones correspondientes a cada capa son bastante similares.

En cambio, estos dos modelos tienen diferencias entre ellos. OSI fue ideado antes de que los protocolos que lo cumplen, por tanto, no está orientado a ningún protocolo en concreto. Por el contrario, los protocolos fueron desarrollados antes que el modelo TCP/IP, siendo este último un modelo para que dichos protocolos encajen a medida haciendo difícil que cuadre para otro grupo de protocolos diferentes.

2.2 DESCRIPCIÓN DEL PROTOCOLO

SNMP fue un protocolo ideado a finales de la década de los 80, por tanto, a lo largo de los años ha ido implementando nuevas versiones con mejoras respecto a sus predecesores. A lo largo de este apartado se irán explicando las diferentes características del protocolo SNMP en función de cómo éstas fueron siendo añadidas a sus diferentes versiones.

2.2.1. SNMP V1

El objetivo inicial del *Network Management* era crear un único protocolo que manejara tanto redes con modelo TCP/IP como redes con modelo OSI. Con este objetivo en mente, SNMP fue inicialmente ideado como un grupo de especificaciones provisionales mientas que el CMOT se suponía como el protocolo que sería finalmente utilizado.

Arquitectura

El modelo de *Network Management* que se utiliza para las redes TCP/IP incluye los siguientes elementos:

-Management station (manager): es donde se albergan las diferentes aplicaciones de *management.*

-Management agent (agente): el elemento que proporciona la información contenida en la MIB a las aplicaciones de *management* y recibe la información de control del manager.

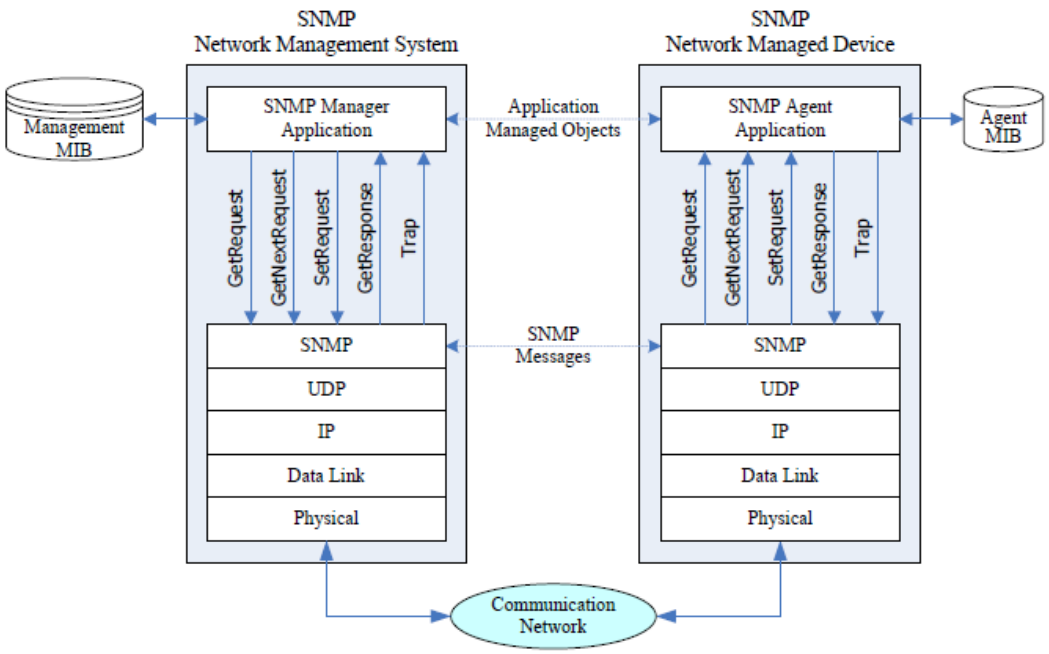
-MIB (Management information base): la información que puede ser obtenida de los agentes y controlada por los managers.

-Protocolo: define las normas para conectar y comunicar al agente con el manager.

La arquitectura de SNMP, detallada en la figura 2.2, muestra los principales elementos del entorno de una red. SNMP está diseñado para ser un protocolo basado en pequeños mensajes para la capa de Aplicación. SNMP es implementado a través de UDP, por lo que tango el manager como el agente deben soportar ambos protocolos, SNMP y UDP.

SNMP es un protocolo sin conexión, es decir, cada intercambio de información entre un agente y un manager es una operación independiente. Esta característica permite minimizar la complejidad de los agentes.

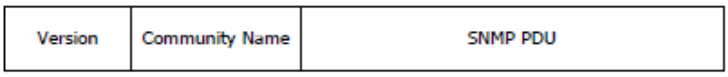
En la figura 2.2, a parte de la arquitectura, también se muestran los 5 tipos de Unidad de datos de protocolo (PDUs por sus siglas en inglés) soportados por SNMP. El manager puede emitir 3 tipos de PDU: GetRequest, GetNextRequest, SetRequest. Los dos primeros son dos variantes diferentes de la función Get. El agente por su parte puede emitir dos PDUs: GetResponse y Trap. El primero como respuesta a la función Get del manager, el segundo como mensaje no pedido que se envía cuando se detecta una anomalía en los diferentes componentes del MIB.



2.3 Arquitectura de SNMP

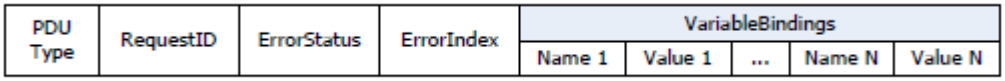
Formato de Mensaje

Los paquetes de datos que se envían con SNMP durante la comunicación entre los Agentes y los Managers. Consisten en 3 partes diferenciadas: el identificador de versión indicando que versión de SNMP se utiliza, el nombre de comunidad utilizado para ese paquete en concreto que sirve como de código para autenticar, y el PDU de SNMP. Esta estructura se puede apreciar en la figura 2.4.



2.4 Mensaje SNMP

Los PDU de SNMP a su vez se estructuran de la siguiente manera:



2.5 estructura PDU de las funciones SET y GET

Los significados de cada campo de los mensajes GET y SET son los siguientes:

Tabla 1. Campos de Mensajes GET y SET de SNMP

Campo	Función
PDU Type	Permite distinguir entre los 5 tipos de mensaje GetRequest (0), GetNextRequest (1), SetRequest (2), GetResponse (3), Trap (4).
Request ID	Permite distinguir entre varios request pendientes, es un identificador único al mensaje.
ErrorStatus	Si es distinto de 0 indica que se ha producido un error al procesar el request.
ErrorIndex	Indica mayor información sobre el posible error ocurrido.
VariableBindings	Lista de nombres de variables y sus valores

Por su lado, los mensajes Trap, al ser algo peculiares al resto de tipos de mensaje, tienen la siguiente estructura:

PDU Type	Enterprise	Agent-Address	Generic-Trap	Specific-Trap	Timestamp	VariableBindings				
						Name 1	Value 1	...	Name N	Value N

2.6 estructura PDU de la función TRAP

Los significados de los campos de los mensajes Trap son los siguientes:

Tabla 2. Campos del mensaje Trap de SNMP

Campo	Función
PDU Type	Permite distinguir entre los 5 tipos de mensaje GetRequest (0), GetNextRequest (1), SetRequest (2), GetResponse (3), Trap (4).
Enterprise	Tipo de objeto generando el Trap.
AgentAddress	Dirección del Agente que general el Trap.
GenericTrap	Tipo de generic trap, los posibles valores son: coldStart (0), warmStart (1), linkDown (2), linkUp (3), authenticationFailure (4), egpNeighborLoss (5), enterpriseSpecific (6)
SpecificTrap	Código especifico del Trap cuando corresponde a un enterptiseSpecific (6)
TimeStamp	Tiempo transcurrido desde la última re-inicialización.
VariableBindings	Lista de nombres de variables y sus valores.

Seguridad

La única opción de seguridad que SNMPv1 proporciona es el ya mencionado nombre de comunidad contenido en los bloques de mensaje SNMP como se aprecia en la figura 2.4. Este dato sirve como “contraseña” para autentificar un mensaje SNMP. Sin ningún tipo de encriptación, esta característica no proporciona apenas

ningún tipo de seguridad ya que puede ser fácilmente obtenido mientras pasa del Agente al Manager. Además, SNMP no puede comprobar la fuente de un mensaje, por ello, es posible que un usuario no autorizado ejerza funciones SNMP para poder obtener información. Por estas deficiencias de seguridad, muchas componentes que soportan SNMP han decidido no permitir la función SET, reduciendo así su capacidad de ser controladas de manera efectiva.

2.2.2. SNMP V2

Como se explicó anteriormente, SNMP fue desarrollado como una solución provisional mientras que CMOT, el cual permite que protocolos de sistemas con modelo OSI puedan ser compatibles con el protocolo TCP, pretendía ser la solución final al *Network Management.* No obstante, esto último nunca se llegó a producir y, al mismo tiempo, SNMP fue extendiéndose requiriendo una mejora del protocolo. SNMPv2 fue entonces desarrollado cuando se vio que SNMP iba a ser el protocolo definitivo.

La arquitectura de SNMPv2 es esencialmente la misma que la de la de su versión anterior, los principales cambios introducidos en esta segunda versión son:

-Capacidad de trasferencia de datos en bloque: El cambio más destacable de SNMPv2 es la inclusión de 2 nuevos PDUs. El primero es GetBulkRequest PDU, que permite al Manager recolectar grandes bloques de datos de manera eficiente y, por tanto, acelerando el proceso GetNextRequest.

-Capacidad Manager-to-Manager: El segundo PDU añadido es InformRequest PDU, que permite a un Manager a enviar informaciones de estilo Trap a otro Manager.

Formato de Mensaje

Para mejorar la eficiencia del intercambio de datos, la estructura de los PDUs se unificó para todos los mensajes como la utilizada para las funciones GET y SET

(Figura 2.5). El único PDU que tiene alguna diferencia con el resto es el recién introducido GetBulkRequest.

PDU Type	RequestID	NonRepeaters	MaxRepetitions	VariableBindings				
				Name 1	Value 1	...	Name N	Value N

2.7 estructura PDU de GetBulkRequest

Los campos diferentes, NonRepeaters indica el número de datos diferentes que se piden en el bloque, y MaxRepetitions indica el máximo de veces que se repite uno de los datos pedidos.

Seguridad

Al ser muy parecido a SNMPv1, SNMPv2 no es capaz de proporcionar ningún servicio de seguridad. La seguridad sigue siendo igual de deficiente en esta segunda versión y los peligros siguen siendo los mismos que los de su predecesor.

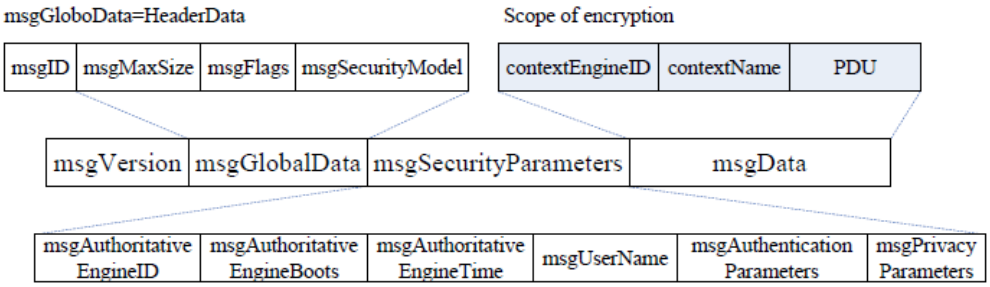
2.2.3. SNMPV3

Las deficiencias en materia de seguridad de las dos primeras versiones hicieron que se desarrollara una tercera y definitiva (hasta el momento) versión de SNMP. La configuración de SNMPv3 puede ser ajustada de manera remota a través de enlaces de comunicación seguros. SNMPv3 también proporciona un marco de referencia para las versiones anteriores de SNMP y para posibles futuros desarrollos sin necesidad de realizar grandes cambios.

Formato de Mensaje

Para mejorar la seguridad, el principal objetivo del SNMPv3, uno de los elementos que más cambios sufrió ha sido la estructura de los mensajes.

Cada mensaje de SNMPv3 incluye 4 data groups: msgVersion, msgGlobalData, msgSecurityParameters y msgPDU como se puede apreciar en la figura 2.8



2.8 Formato de mensaje de SNMPv3

El apartado msgVersion se ajusta a snmpv3(3) e identifica el mensaje como uno de la versión 3 de SNMP. El bloque msgGlobalData contiene la información del header. El bloque msgSecurityParameters es usado exclusivamente por el modelo de seguridad. El bloque de msgData es el principal donde se encuentra el PDU y la información de identificación para distinguir este PDU.

Seguridad

Este es la principal razón por la que existe un SNMPv3, por ello es en este apartado donde se producen más modificaciones. SNMPv3 intenta resolver 4 tipos de peligros de seguridad: modificación de información, enmascaramiento de usuarios, revelación de información y modificación del flujo de mensajes; siendo los dos primeros los considerados como los principales peligros.

Para ello se diseña un modelo de seguridad basado en el usuario (USM) el cual refleja el concepto tradicional de que un usuario se identifique por un "username".

USM establece que cuando dos entidades se comuniquen, una de los dos sea designado como "miembro SNMP autorizado" y que sea ella la que controle la seguridad, evitando así la repetición de mensajes y el retraso producido por ello.

A su vez, USM propone dos procesos, el primero, el de **autenticación** en el que usa dos protocolos alternativos para asegurar la integridad de los datos y la comprobación de su origen: HMAC-MD5-96 y HMAC-SHA-96. HMAC utiliza una función *hash* segura y una clave secreta para producir un código de mensaje de autenticación. El segundo, de **encriptación,** USM utiliza el protocolo de encriptación simétrica CBC-DES para evitar la revelación del contenido de un mensaje.

2.3 MIB

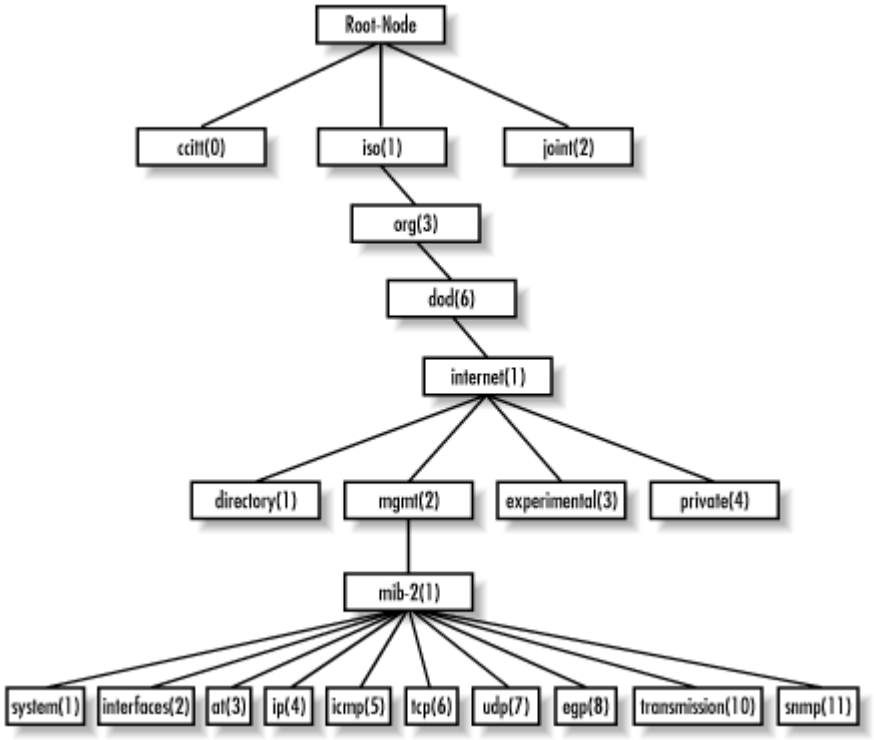
Los cimientos de un Network Management System es una base de información gestionada MIB que contenga los objetos que sean controlados. La MIB es una base de datos estructurada en forma de árbol. Cada sistema en una red contiene una MIB en la que se almacenan los valores de las variables a ser controladas en ese sistema. Esa información puede ser usada por un Manager para poder monitorizar y controlar una entidad de la red. La SMI o estructura de la información gestionada, define la sintaxis y la semántica utilizada para describir la información gestionada en SNMP.

Estructura del MIB

Por simplicidad, la SMI evita tipos complejos de datos. Cada tipo de objeto en una MIB tiene un nombre, sintaxis, y un esquema de codificación. Cada objeto es identificado de manera única por un Object Identifier (OID). La sintaxis de un objeto viene definida por ASN.1 y el esquema de codificación viene definido por el BER.

Los objetos tienen estructura de árbol. Empezando por la raíz del árbol OID, cada valor representa una "rama" del árbol. La raíz tiene tres nodos: el más habitual para SNMP iso (1), itu (0) y joint-iso-itu (2). Se puede observar algunos de los nodos en la figura 2.9.

El OID se conforma de una serie de números separados por un punto que definen el camino tomado en el árbol. Así, por ejemplo, el nodo de internet sería el 1.3.6.1. SMI define 4 nodos por debajo de internet: directory, mgmt, experimental y private. El subárbol de mgmt contiene las definiciones de MIBs que han sido aprobado por la IAB. Dos versiones de MIBs con el mismo OID (1.3.6.1.2.1) han sido desarrolladas MIB-1 y MIB-2.



2.9 árbol OID

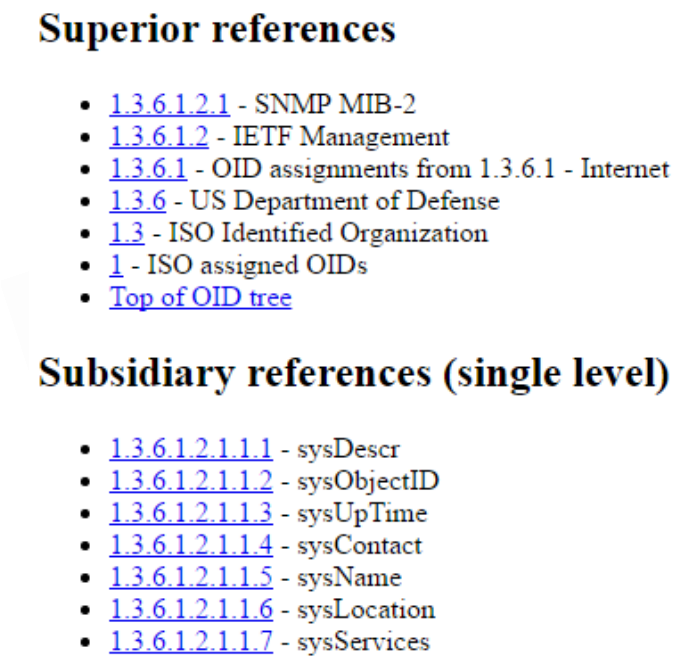
Los objetos contenidos en esa MIB se pueden ver en la siguiente tabla:

Tabla 3 OIDs contenidos en MIB-2

Grupo	Descripción
system	Contiene descripción del sistema e información administrativa
interfaces	Contiene información sobre cada interfaz del sistema a una subred.
at	Contiene la tabla de traducción de direcciones internet-to-subred.
ip	Contiene la información relevante a la implementación y operación de IP en el nodo.
icmp	Contiene la información relevante a la implementación y operación de ICMP en el nodo.

tcp	Contiene la información relevante a la implementación y operación de TCP en el nodo.
udp	Contiene la información relevante a la implementación y operación de UDP en el nodo.
egp	Contiene la información relevante a la implementación y operación de EGP en el nodo.
transmission	Contiene información sobre los esquemas de transmisión y el acceso a protocolos en cada interfaz del sistema.
snmp	Contiene la información relevante a la implementación y operación de SNMP en este sistema.

A su vez, cada uno de estos objetos tienen una serie de variables como se puede ver en la figura 2.10 que son las que serán después probadas en la parte práctica.

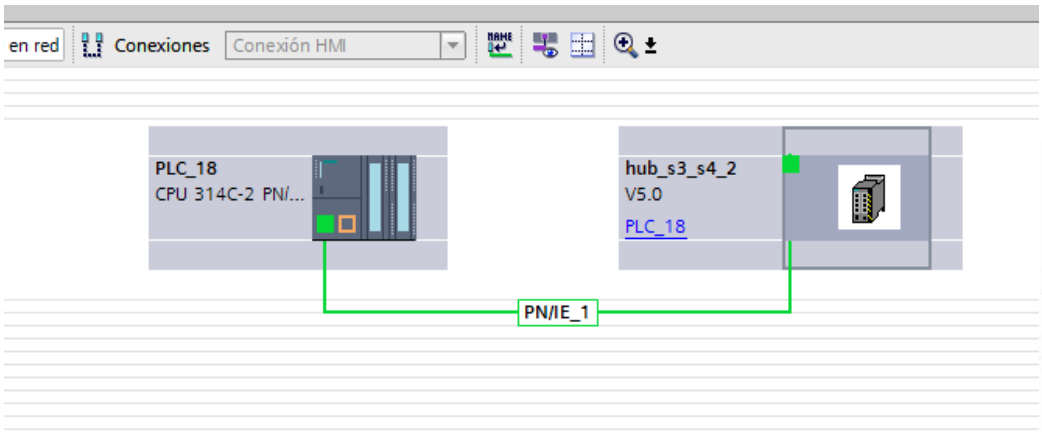


2.10 Camino del árbol del OID 1.3.6.1.2.1 system (arriba) y variables contenidas en él (abajo)

2.4 IMPLEMENTACIÓN EN EL LABORATORIO

En el caso de SNMP, las pruebas se realizaron en el laboratorio de Automatización y Robótica (LSD1) ya que en él se contaba con los componentes necesarios para poder ponerlo a prueba.

Por un lado, tenemos el hardware: switches y PLCs de la marca Siemens que formaban en el laboratorio una red de comunicación como las que se pretendía estudiar y que, como la mayoría de aparatos de este estilo en el mercado, estaban preparados para soportar SNMP. Por otro lado, tenemos el software, el programa TIA Portal (SIMATIC Step7) que permite el control de los PLCs y que, junto con la librería desarrollada de Siemens, permite el control de SNMP.



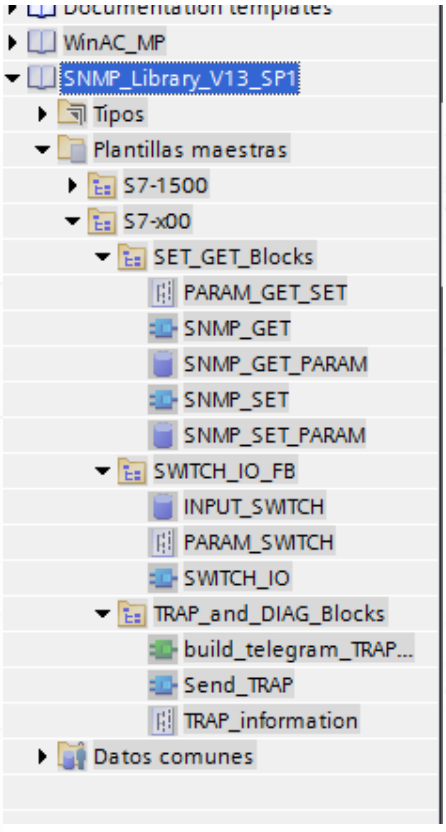
2.11 Conexión de PLC y switch en TIA Portal

Lo primero que se hizo en el TIA Portal fue conectar los elementos como se muestra en la figura 2.11, se utilizó un PLC como los utilizados habitualmente en las prácticas del laboratorio del modelo S7 314C-2 PN/DP. El elemento a controlar con SNMP es un Switch modelo SCALANCE-X208, para su representación al TIA Portal hizo falta la descarga y adición del archivo GSDML a través del siguiente link ya que los switches no estaban incluidos en los posibles dispositivos a añadir a la red. Una vez añadido el GSDML se añadió la configuración necesaria de ambos componentes (IP, nombre PROFINET etc.) se probó la conexión de ambos componentes como se ve en la figura 2.12.

2.12 Configuración y prueba de conexión PLC y Switch

Una vez establecida la conexión se añadió la librería SNMP, descargada del siguiente link, con la que poder trabajar con SNMP propiamente dicho sobre el PLC y el Switch.

La librería descargada proporciona una serie de data blocks y function blocks necesarias para poder programar el PLC y poder ejecutar las diferentes funciones deseadas de SNMP. La librería proporciona bloques y funciones tanto para CPUs S7-1500 como para S7-x00 (nuestro caso en el ejemplo). En la misma página de descarga de la librería, se adjunta una guía de usuario con la descripción de los diferentes data blocks y function blocks de la librería, explicando el significado de las diferentes variables y datos con los que hay que rellenar los bloques.

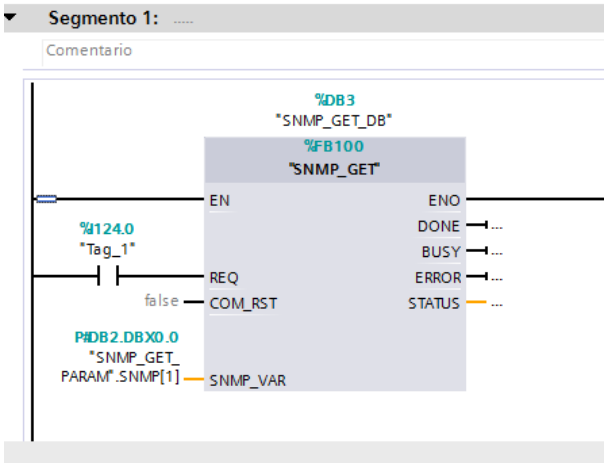


2.13 librería SNMP

GET

La primera función a probar es el comando GET que proporciona los PDUs mencionados anteriormente en la teoría GETRequest y GetResponse además del GetNextRequest.

Con esta función podremos obtener los diferentes valores de varias variables del Switch como su IP, su Tiempo en funcionamiento, su nombre en la red o los servicios que desempeña. Para ello debemos utilizar el function block SNMP_GET de la librería y el data block SNMP_GET_PARAM.



2.14 SNMP GET block

Como podemos observar en la figura 2.14 el bloque GET tiene dos entradas principales que hay que rellenar, una es el REQ, al cual se le coloca un interruptor del panel (en este caso el 124.0) con el que poder disparar la función al activarlo. La otra entrada, y principal fuente de información del bloque es la SNMP_VAR. En esta entrada se coloca todas las variables necesarias para el Request. Esta entrada será ocupada por el data block SNMP_GET_PARAM donde se incluirá toda esta información y, además, se podrán leer los datos respondidos por el agente, el Switch en este caso.

SNMP[1]	"PARAM_GET_SET"	0.0	
IP_ADR	DWord	0.0	16#C0A83801
DEVICE_ID	Byte	4.0	2
CON_ID	Word	6.0	W#16#62
LOCAL_P	Word	8.0	W#16#7D0
OID	String[254]	10.0	'1.3.6.1.2.1.4.3.0'
Community	String[20]	266.0	'private'
VALUE_TYPE	Byte	288.0	16#0
VALUE_LEN	Byte	289.0	16#0
VALUE	Array[1..255] of Byte	290.0	
VALUE[1]	Byte	0.0	16#0
VALUE[2]	Byte	1.0	16#0
VALUE[3]	Byte	2.0	16#0
VALUE[4]	Byte	3.0	16#0
VALUE[5]	Byte	4.0	16#0
VALUE[6]	Byte	5.0	16#0
VALUE[7]	Byte	6.0	16#0

2.15 SNMP_GET_PARAM

Para el bloque GET solo hace falta definir 4 parámetros de esa lista. El primero es la IP_ADR del agente, en este caso el Switch, 192.168.56.1 que traducido al hexadecimal se convierte en C0 A8 38 01. El device_ID se obtiene de la máscara subred, para los CPU S7-300 suele ser el 2 siempre. Tanto el CON_ID como el LOCAL_P se mantienen invariantes, ya que representan los puertos y conexiones de UDP. El OID es una de las claves, ya que, como se explica en la parte teórica, representa el "código" de la variable a observar, este código viene estandarizado según la MIB y por ello, tanto al Agente como el Manager entienden a qué se refiere. En este caso en concreto la variable a observar 1.3.6.1.2.1.4.3.0 representa IpInReceives, que quiere decir el "número total de datagramas recibido por interfaces, incluyendo errores". Por último, hay que rellenar el Community, que puede ser private o public y simplemente tiene que coincidir con el establecido en la configuración del Switch. El resto de variables son rellenadas por la respuesta que envía el Agente con la información, como se observa en la siguiente figura.

Static						
SNMP	Array[1..4] o...	0.0				
SNMP[1]	"PARAM_GET_SET"	0.0				
IP_ADR	DWord	0.0	16#C0A83801	16#C0A8_3801		
DEVICE_ID	Byte	4.0	2	16#02		
CON_ID	Word	6.0	W#16#62	16#0062		
LOCAL_P	Word	8.0	W#16#7D0	16#07D0		
OID	String[254]	10.0	'1.3.6.1.2.1.4.3.0'	'1.3.6.1.2.1.4.3.0'		
Community	String[20]	266.0	'private'	'private'		
VALUE_TYPE	Byte	288.0	16#0	16#41		
VALUE_LEN	Byte	289.0	16#0	16#04		
VALUE	Array[1..255] of Byte	290.0				
VALUE[1]	Byte	0.0	16#0	16#00		
VALUE[2]	Byte	1.0	16#0	16#8A		
VALUE[3]	Byte	2.0	16#0	16#09		
VALUE[4]	Byte	3.0	16#0	16#86		
VALUE[5]	Byte	4.0	16#0	16#00		
VALUE[6]	Byte	5.0	16#0	16#00		
VALUE[7]	Byte	6.0	16#0	16#00		

2.16 SNMP Respuesta a GET de IpInReceives

Una vez cargado y ejecutado el programa, se ve la respuesta enviada por el Switch. Primero se indica el tipo del valor, en este caso 41, que quiere decir contador, indicando que es una variable que va en aumento según ocurran más casos. Se explicará más adelante, los 4 diferentes tipos de variables existentes. Después, nos indica la longitud del dato en la variable VALUE_LEN, en Bytes, en este caso 4. Por último, en la variable VALUE, nos da el dato en sí, dato en hexadecimal que hay que saber traducir dependiendo del tipo de dato especificado por VALUE_TYPE. En este caso, al tratarse de un contador, simplemente hay que traducir el dato a número decimal del hexadecimal, 8A0986 sería equivalente a 9046406 datagramas totales recibidos por este Switch.

Para entender mejor los diferentes tipos de variables y comprobar su funcionamiento correcto, se probaron otras variables para obtener diferentes datos de diferentes tipos. En la figura 2.17 se puede apreciar el GETRequest de la variable SysName que no es más que el nombre asignado a este nodo, una variable que es modificable y que luego será la que se utilice en el comando SET. En este caso se puede apreciar que el VALUE_TYPE es 04, indicando que es una string de texto, y su longitud es 8 bytes. Teniendo eso en cuenta el resultado obtenido 72 6F 6F 6D

2D 30 31 35 se puede traducir por el nombre asignado al nodo que en este caso es "room-015".

SNMP[2]	"PARAM_GET_SET"	546.0			
IP_ADR	DWord	0.0	16#C0A83801	16#C0A8_3801	
DEVICE_ID	Byte	4.0	2	16#02	
CON_ID	Word	6.0	W#16#0064	16#0064	
LOCAL_P	Word	8.0	W#16#7D1	16#07D1	
OID	String[254]	10.0	'1.3.6.1.2.1.1.5.0'	'1.3.6.1.2.1.1.5.0'	
Community	String[20]	266.0	'private'	'private'	
VALUE_TYPE	Byte	288.0	16#0	16#04	
VALUE_LEN	Byte	289.0	16#0	16#08	
VALUE	Array[1..255] of Byte	290.0			
VALUE[1]	Byte	0.0	16#0	16#72	
VALUE[2]	Byte	1.0	16#0	16#6F	
VALUE[3]	Byte	2.0	16#0	16#6F	
VALUE[4]	Byte	3.0	16#0	16#6D	
VALUE[5]	Byte	4.0	16#0	16#2D	
VALUE[6]	Byte	5.0	16#0	16#30	
VALUE[7]	Byte	6.0	16#0	16#31	
VALUE[8]	Byte	7.0	16#0	16#35	
VALUE[9]	Byte	8.0	16#0	16#00	
VALUE[10]	Byte	9.0	16#0	16#00	

2.17 SNMP Respuesta a GET de sysName

La siguiente prueba, como se puede apreciar en la figura 2.18, se hizo sobre la variable 1.3.6.1.2.1.1.3.0 que corresponde a sysUpTime, es decir, el tiempo que lleva este componente conectado a la red desde su última re-inicialización.

SNMP[2]	"PARAM_GET_SET"	546.0		
IP_ADR	DWord	0.0	16#C0A83801	16#C0A8_3801
DEVICE_ID	Byte	4.0	2	16#02
CON_ID	Word	6.0	W#16#0064	16#0064
LOCAL_P	Word	8.0	W#16#7D1	16#07D1
OID	String[254]	10.0	'1.3.6.1.2.1.1.3.0'	'1.3.6.1.2.1.1.3.0'
Community	String[20]	266.0	'private'	'private'
VALUE_TYPE	Byte	288.0	16#0	16#43
VALUE_LEN	Byte	289.0	16#0	16#04
VALUE	Array[1..255] of Byte	290.0		
VALUE[1]	Byte	0.0	16#0	16#35
VALUE[2]	Byte	1.0	16#0	16#D8
VALUE[3]	Byte	2.0	16#0	16#36
VALUE[4]	Byte	3.0	16#0	16#D6
VALUE[5]	Byte	4.0	16#0	16#00
VALUE[6]	Byte	5.0	16#0	16#00

2.18 SNMP Respuesta a GET de sysTimeUp

Este Request devolvió un VALUE_TYPE de 43, que es un valor especifico de SNMP denominado TimeTicks. Corresponde a un contador de segundos en centésimas de segundo. En este caso el valor obtenido es de 35D836D6, es decir, 903362262 centésimas de segundo, lo que equivale a un valor aproximado de 104.5 días conectado desde su última re-inicialización.

Por último, y para terminar con los tipos de datos se probó la variable 1.3.6.1.2.1.1.7.0 (Figura 2.18) que corresponde a Sys Services, una variable que indica los servicios que este Switch desempeña.

_GET_SET"	546.0		
	0.0	16#C0A83801	16#C0A8_3801
	4.0	2	16#02
	6.0	W#16#0064	16#0064
	8.0	W#16#7D1	16#07D1
54]	10.0	'1.3.6.1.2.1.1.7.0'	'1.3.6.1.2.1.1.7.0'
0]	266.0	'public'	'public'
	288.0	16#0	16#02
	289.0	16#0	16#01
.255] of Byte	290.0		
	0.0	16#0	16#43
	1.0	16#0	16#00
	2.0	16#0	16#00
	3.0	16#0	16#00
	4.0	16#0	16#00
	5.0	16#0	16#00

2.19 SNMP Respuesta a GET de sysServices

Esta variable en concreto devuelve un único número entero (VALUE_TYPE 02) que sirve como código para indicar las actividades que este nodo desempeña. El numero devuelto corresponde a una suma de potencias diferentes de 2. La suma empieza en 0, y va añadiendo, para cada capa en la que este nodo realiza una acción 2^(C-1) siendo los códigos de las capas los siguientes:

1 física (por ejemplo, repetidores)

2 enlace de datos/subred (por ejemplo, puentes de red)

3 internet (IP gateway)

4 end-to-end (IP host)

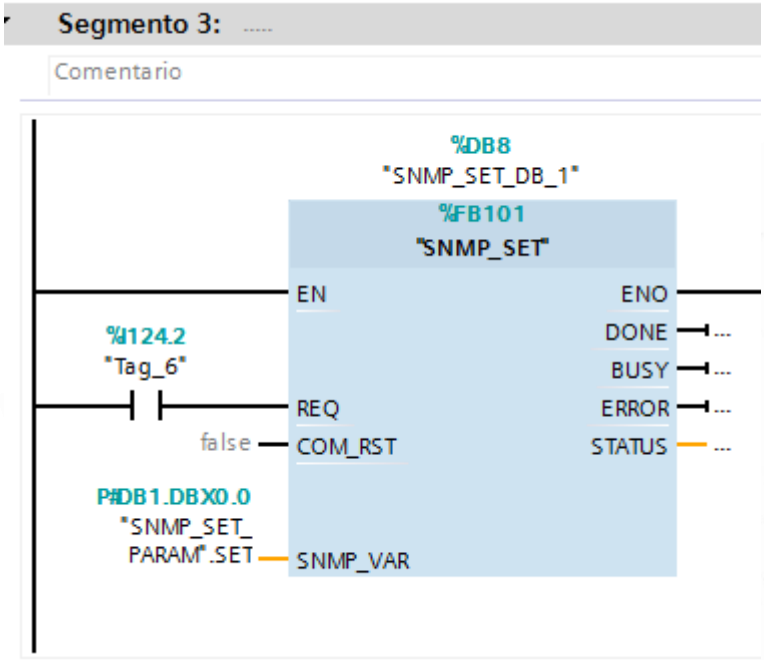
7 aplicaciones

Por tanto, un nodo que devuelve el valor 72, por ejemplo, 72= (2^(4-1) +2^(7-1)) por lo que participa en las capas 4 (end-to-end) y 7 (aplicación).

En el caso observado en el laboratorio, el Switch devuelve el valor hexadecimal 43, que equivale al 67. 67 se descompone en 2^(7-1), 2^(2-1) y 2^(1-1) por lo que, con este dato, se puede hallar que el Switch participa en las capas de Aplicación (7), Enlace de datos (2) y física (1).

SET

La siguiente función a probar en fue la función SET equivalente al SetRequest mencionado anteriormente en la teoría, esta función permite asignar el valor que se quiera a una variable concreta del Switch.



2.20 Bloque Set

El bloque SET es muy similar al Get, se debe colocar un interruptor en REQ para poder también accionar la función (124.2) y en SNMP_VAR se ha de incluir una variable muy similar a la de GET, pero en este caso SNMP_SET_PARAM.

Static			
SET	"PARAM_GET_SET"	0.0	
IP_ADR	DWord	0.0	16#C0A83801
DEVICE_ID	Byte	4.0	2
CON_ID	Word	6.0	W#16#0062
LOCAL_P	Word	8.0	W#16#07D0
OID	String[254]	10.0	'1.3.6.1.2.1.1.5.0'
Community	String[20]	266.0	'private'
VALUE_TYPE	Byte	288.0	16#04
VALUE_LEN	Byte	289.0	16#08
VALUE	Array[1..255] of Byte	290.0	
VALUE[1]	Byte	0.0	16#72
VALUE[2]	Byte	1.0	16#6F
VALUE[3]	Byte	2.0	16#6F
VALUE[4]	Byte	3.0	16#6D
VALUE[5]	Byte	4.0	16#2D
VALUE[6]	Byte	5.0	16#51
VALUE[7]	Byte	6.0	16#FA
VALUE[8]	Byte	7.0	16#34
VALUE[9]	Byte	8.0	16#0
VALUE[10]	Byte	9.0	16#0

2.21 SNMP_SET_PARAM

Como se puede apreciar en la figura, el data block tiene la misma estructura que el de GET, la diferencia es que, en este caso, al asignar una variable, aparte de IP_ADR, DEVICE_ID, OID y Community como se rellenaron en el bloque GET, se debe designar el VALUE_TYPE, VALUE_LEN y VALUE. Para ello, se debe saber qué tipo de valores se ha de añadir 02(numero entero), 04(string de texto), 41(contador) o 43(Timeticks).

Lo primero que se tiene que saber es si la variable permite ser modificada, existe una serie de OIDs que como Access tiene Read-only y que por tanto no permiten ser utilizadas con el comando SET, otras, en cambio, como Access tienen Read-Write permitiendo tanto el acceso por GET como la modificación a través de SET.

Es recomendable recurrir a la base de datos http://www.alvestrand.no/objectid/top.html donde se puede buscar cualquier OID

registrado. En ella, se especifica de cada OID el nombre de la variable, en que consiste exactamente, el tipo de variable que es (integer, string, counter o Timetick) y su acceso. Con esta información se puede saber si una variable es modificable y que parámetros se deben introducir en SNMP_SET_PARAM para su correcto funcionamiento.

Para el ejemplo concreto se utilizará la variable 1.3.6.1.2.1.1.5 sysName que tiene un Acceso Read-Write y por tanto puede ser modificada. Se introduce el VALUE_TYPE 04 al tratarse de un string de texto y el VALUE_LEN acorde a la longitud. Recordemos que el valor de esa variable que fue una de las probadas con el comando GET era de 72 6F 6F 6D 2D 30 31 35 que se traduce por “room-015”.

Para esta prueba cambiaremos los últimos 3 valores siendo la nueva variable 72 6F 6F 6D 2D 51 FA 34 como se aprecia en la figura 2.21.

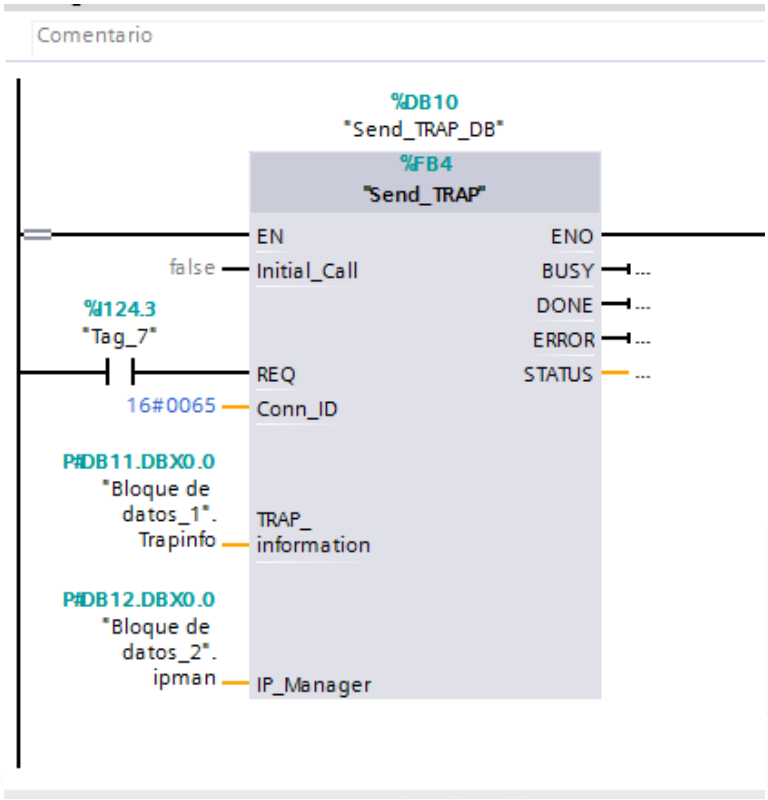
SNMP[2]	"PARAM_GET_SE...	546.0		
IP_ADR	DWord	0.0	16#C0A83801	16#C0A8_3801
DEVICE_ID	Byte	4.0	2	16#02
CON_ID	Word	6.0	W#16#0064	16#0064
LOCAL_P	Word	8.0	W#16#7D1	16#07D1
OID	String[254]	10.0	'1.3.6.1.2.1.1.5.0'	'1.3.6.1.2.1.1.5.0'
Community	String[20]	266.0	'private'	'private'
VALUE_TYPE	Byte	288.0	16#0	16#04
VALUE_LEN	Byte	289.0	16#0	16#08
VALUE	Array[1..255] of Byte	290.0		
VALUE[1]	Byte	0.0	16#0	16#72
VALUE[2]	Byte	1.0	16#0	16#6F
VALUE[3]	Byte	2.0	16#0	16#6F
VALUE[4]	Byte	3.0	16#0	16#6D
VALUE[5]	Byte	4.0	16#0	16#2D
VALUE[6]	Byte	5.0	16#0	16#51
VALUE[7]	Byte	6.0	16#0	16#FA
VALUE[8]	Byte	7.0	16#0	16#34
VALUE[9]	Byte	8.0	16#0	16#00

2.22 SNMP GET tras haber modificado sysName

Una vez se modifica con SET, se realiza un GET de nuevo a la variable sysName y, como se puede apreciar en la figura 2.22, los valores obtenidos son, en efecto, los asignados con SET: 72 6F 6F 6D 2D 51 FA 34 que pueden ser traducidos por el siguiente texto: “room-Qú4” y es el nuevo sysName del Switch.

TRAP

Para concluir, la última función destacable de SNMP es la TRAP, la función TRAP, como se ha explicado en la teoría, es la única función en la que el Manager no envía una petición. La función se comporta como una "baliza" colocada en el Agente y que observa a una variable en concreto, el Agente envía el resultado periódicamente al Manager con el valor de dicha variable.



2.23 Bloque Trap

El bloque Trap es diferente a los GET y SET, ya que requiere más información, como se puede apreciar en la figura 2.24, a parte de los ya clásicos IP del Agente y OID de la variable, se ha de incluir otros parámetros nuevos como la IP del Manager a la que enviar el Trap, el mensaje de dicho Trap (Telegram). Toda esta información se incluye en el data block TRAP_INFO.

Bloque de datos_1

	Nombre	Tipo de datos	Offset	Valor de arranq...	Remanen...	Visi
1	Static				☐	
2	Trapinfo	"TRAP_information"	0.0		☑	
3	IP_agent	"IP_addr"	0.0		☑	
4	Addr	Array [0 .. 3] of Byte	0.0		☑	
5	Addr[0]	Byte	0.0	16#C0	☑	
6	Addr[1]	Byte	1.0	16#A8	☑	
7	Addr[2]	Byte	2.0	16#38	☑	
8	Addr[3]	Byte	3.0	16#01	☑	
9	TrapID	String[125]	4.0	'1.3.6.1.2.1.1.3.0'	☑	
10	specificID	Int	132.0	2	☑	
11	OID_BindingVar	String[125]	134.0	'1.3.6.1.2.1.1.3.0'	☑	
12	Description	String[125]	262.0	''	☑	
13	Telegram	"TRAP_telegram"	390.0		☑	
14	<Agregar>				☐	

2.24 SNMP data block TrapInfo

Una vez cargada la función TRAP en el PLC no fue posible recibir los resultados, la función SNMP_TRAP_Telegram disponible en la librería, la cual se encarga de generar el telegrama con la información de la Trap recibida, no era compatible con la versión disponible en el laboratorio de TIA Portal y no permitía siquiera su compilación.

2.5 CONCLUSIONES SNMP

Una vez estudiado de forma teórica y probado en el laboratorio de manera práctica las diferentes funciones se pueden destacar las siguientes características del protocolo SNMP:

En primer lugar, cabe destacar que es un protocolo muy clásico, fue desarrollado a finales de los años 80 con una función muy concreta, dar solución al problema del Network Management. Se han ido mejorando y desarrollando a lo largo del tiempo, pero su versión más moderna y vigente en la actualidad es de hace 12 años (SNMPv3 de 2004), por todo ello, es posible que quede un poco restringido a las nuevas tecnologías y a las necesidades que pueda requerir las comunicaciones M2M de hoy en día.

A su favor cuenta con lo extendido del mismo, es un protocolo que es la referencia para el Network Management desde hace más de 20 años y por tanto la mayoría de componentes para redes de comunicación en el mercado y en las instalaciones ya existentes están preparados para soportarlo.

Por todo ello, si lo que se desea es exclusivamente el Network Management de una red ya existente, probablemente SNMP sea la opción más clara y fácil de usar, ya que viene implementada de serie en la mayoría de componentes.

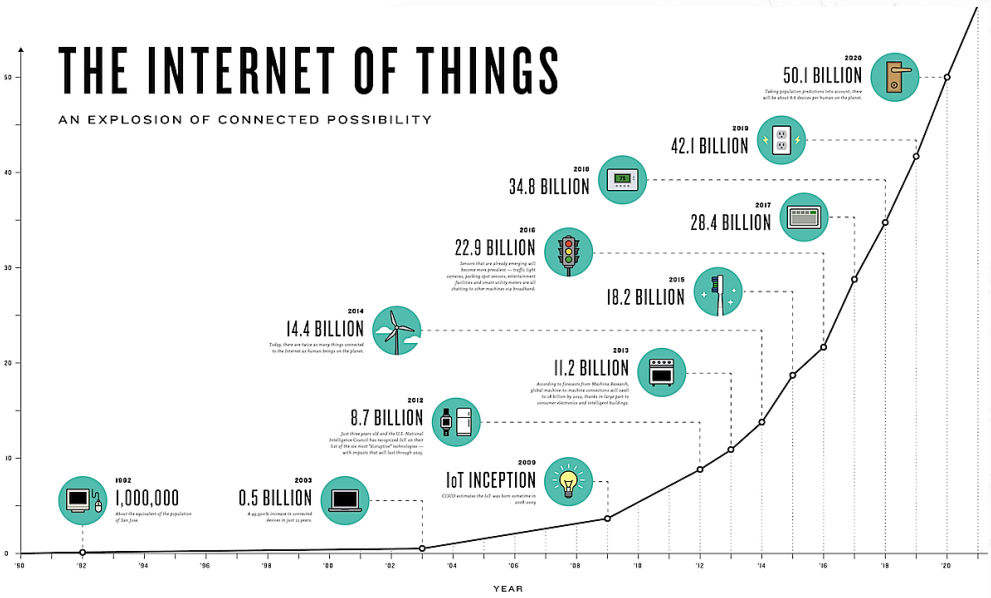
Capítulo 3 CoAP

3.1 *Internet Of Things*

Definición e historia

Internet of Things (IoT) es el término utilizado para describir a la red formada por todo tipo de dispositivos físicos con componentes electrónicos y software que les permite intercambiar entre información entre ellos. Estos dispositivos pueden abarcar desde pequeños aparatos como móviles, sensores o relojes a vehículos, plantas de producción, edificios. En 2013 la Iniciativa de estándares globales para Internet of Things (IoT-GSI) definió el IoT como "la infraestructura de la sociedad de la información".

IoT permite a todo tipo de objetos ser controlados de forma remota, creando una mayor integración del mundo real en sistemas digitales. La mayor expresión de IoT son los sistemas ciberfísicos tales como las smart grids, smart homes o el transporte inteligente.



3.1 Proyección de crecimiento del Internet of Things

IoT es un concepto muy reciente, el primer uso que se le conoce al término Internet of Things fue Kevin Ashton en 1999 mientras trabajaba en los laboratorios Auto-ID del MIT. Su tecnología empezó a desarrollarse a principios del siglo XXI, pero el gran "boom" del concepto y el gran desarrollo de sus tecnologías se ha comenzado a ver en la segunda década de este siglo. Como se puede ver en la figura 3.1, su crecimiento es prácticamente de manera exponencial. Se estima que para 2020 haya unos 50 mil millones de dispositivos conectados a la red, más del doble de los conectados en la actualidad (2016).

Direccionamiento único de los objetos

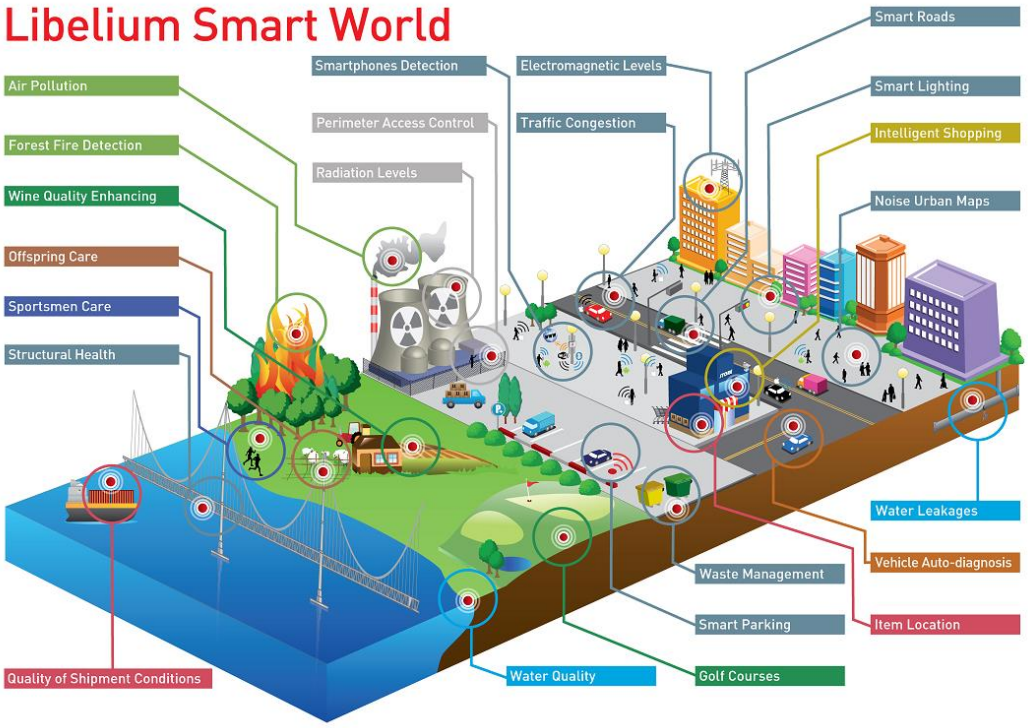
Cada objeto debe ser identificado de manera única. Inicialmente, la idea de Auto-ID era utilizar un código electrónico de producto (EPC) aunque esto evolucionó rápidamente a que los objetos tuvieran una dirección IP o un URI. No obstante, la versión más extendida del protocolo IP, IPv4, se quedaba corto ya que solo contaba con cerca 4.300 millones de direcciones de IP diferentes, número ya superado a día de hoy. Por el contrario, la nueva generación del protocolo IP (IPv6), que a día de hoy está en proceso de popularización y sustitución paulatina de su predecesor, será la solución definitiva al nombramiento de objetos ya que éste permite 340 sextillones o 2^{128} diferentes direcciones, lo que equivale a 670 mil millones de direcciones por milímetro cuadrado de la Tierra.

Aplicaciones

IoT está por todos lados, prácticamente todo campo existente tiene alguna aplicación posible, esto son algunos de los ejemplos:

- Dispositivos electrónicos: la mayoría de dispositivos existentes ya están conectados a la red: smart phones, tablets, ordenadores, smart glasses, smart watches.
- Medio ambiente: sistemas de protección medioambiental como sensores para medir la calidad del agua o condiciones atmosféricas, seguimiento y control de animales salvajes, avisos de emergencia de terremotos o tsunamis.

- Medicina: dispositivos que permitan la vigilancia de la salud a distancia y de forma constante como medidores de presión en sangre, marcapasos, medidores de azúcar, pulseras de control del ritmo cardíaco etc.
- Construcción: construcción de smart homes y smart buildings que controlen todos los sistemas mecánicos, eléctricos y electrónicos de un edificio.
- Transporte: controles de tráfico inteligentes para controlar el flujo de vehículos en cada momento, semáforos, peajes y parkings inteligentes, control de conducción de vehículos o asistencia en carretera.
- Energía: optimización del consumo de energía, smart grids, smart cities.



3.2 Smart world con implementación de IoT

Características

Internet of Things requiere de una gran escalabilidad debido al gran número de dispositivos que se conectan cada instante. IPv6 será el principal responsable de encargarse de la escalabilidad en la capa de aplicación y, en concreto, 6LoWPAN permitirá a pequeños dispositivos con pocos recursos conectarse a redes IP. En la capa de aplicación, protocolos como CoAP y MQTT permitirán la trasmisión de

datos de poco tamaño para poder así facilitar la comunicación a dispositivos con poca memoria.

Seguridad

A día de hoy, la seguridad es uno de los principales retos del IoT. Críticos afirman que está siendo desarrollado de manera muy rápida sin prestar atención a los retos en materia de seguridad que IoT conlleva. En una encuesta en Bussiness Insider en 2014, el 39% de los encuestados afirmaba que su mayor preocupación al utilizar tecnología IoT era la seguridad.

Como respuesta a la preocupación, se creó, por iniciativa de las principales compañías de telecomunicaciones mundiales (BT, Vodafone), la Internet of Things Security Fundation (IoTSF) en septiembre de 2015 con el objetivo de mantener seguro el IoT y promover la buena práctica.

3.2 DESCRIPCIÓN DEL PROTOCOLO

Introducción

Constrained Application Protocol (CoAP) es un protocolo de web especializado para el uso de nodos y redes con limitaciones como bajo consumo de energía, muchas pérdidas o poca memoria. Estos nodos normalmente tienen microcontroladores de 8 bits con poca cantidad de ROM y RAM. El protocolo está diseñado para las aplicaciones M2M como la automatización de edificios o smart energy.

Uno de los principales objetivos de CoAP es diseñar un protocolo genérico para las redes y nodos con limitaciones. Redes como 6LoWPAN permite la fragmentación de paquetes IPv6, no obstante, esto causa una gran reducción el a probabilidad de entrega del paquete. Uno de los objetivos de CoAP es mantener el tamaño de los mensajes evitando la necesidad de fragmentación. El objetivo de CoAP no es simplemente comprimir HTTP, del que toma muchas características, sino utilizar un subconjunto de REST común con HTTP, pero optimizado para aplicaciones M2M.

Las principales características que CoAP aporta son las siguientes:

-Protocolo de Internet que cumpla los requisitos para comunicación M2M en redes limitadas.

-Soporte de UDP con fiabilidad opcional pata unidifusión o multidifusión.

-Intercambios de mensajes de forma asíncrona.

-Cabecera de paquetes de poco tamaño y complejidad.

-Soporte para URI y Content type.

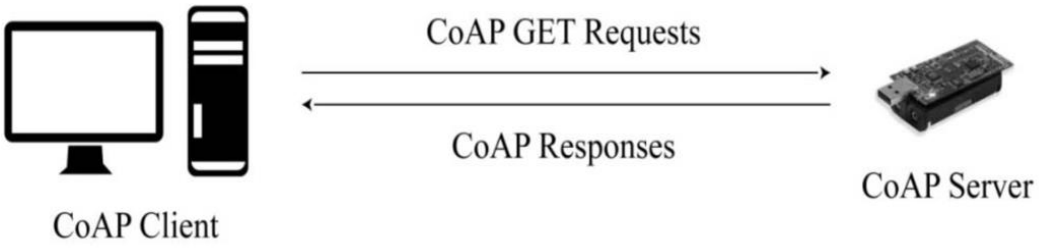
-Capacidad de cache y de proxy simple.

-Mapeo para HTTP que permita a proxies acceso a través de HTTP.

-Seguridad a través de DTLS.

Modelo de comunicación

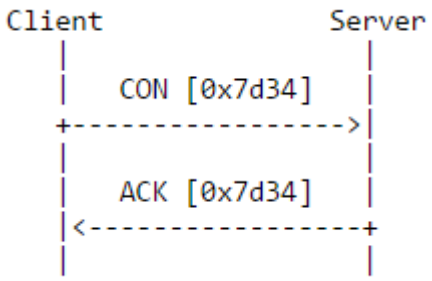
La forma de interacción en CoAP es muy similar al modelo cliente/servidor de HTTP, aunque en interacciones M2M normalmente un elemento CoAP acaba actuando tanto de cliente como de servidor. Un request es equivalente al de HTTP en el que un cliente pide una acción a una fuente, que se identifica por un URI, en un servidor. El servidor contesta enviando una respuesta con un Código de Respuesta. A diferencia de HTTP, CoAP realiza estos intercambios de manera asíncrona a través de UDP.



3.3 Modelo cliente servidor

CoAP define 4 tipos de mensajes: Confirmable (CON), Non-confirmable (NON), Acknowledgement (ACK) y Reset (RST). Los request pueden ser tanto CON como NON y las respuestas CON, NON o acarreadas (piggybacked) en un mensaje ACK.

La fiabilidad se consigue marcando un mensaje como CON. El cual es retrasmitido usando un timeout y exponential back-off entre retransmisiones hasta recibir un ACK del receptor con el mismo Message ID. En la figura 3.4 podemos ver un ejemplo con el Message ID 0x7d34.

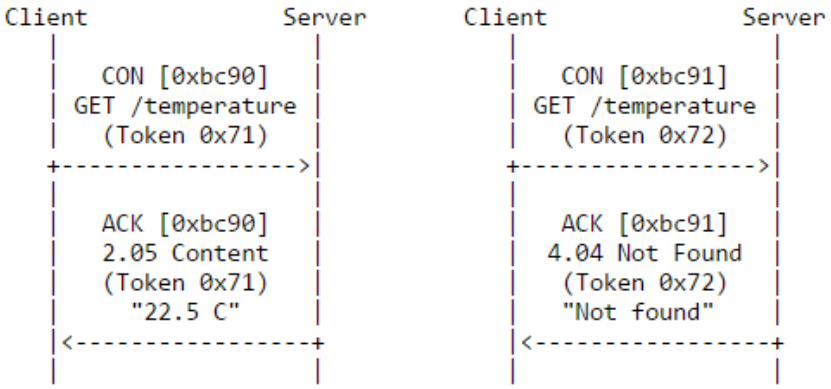


3.4 Transmisión CoAP con fiabilidad

Si el receptor no es capaz de procesar el mensaje recibido responde con un RST en vez de un ACK.

Otra opción son los mensajes que no requieren fiabilidad, como las continuas lecturas de un sensor, y estos pueden ser enviados como NON. Estos mensajes no

son contestados con un ACK, pero aun así utilizan el Message ID para detectar duplicados. También en este caso si el receptor es incapaz de procesar el mensaje responde con un RST.

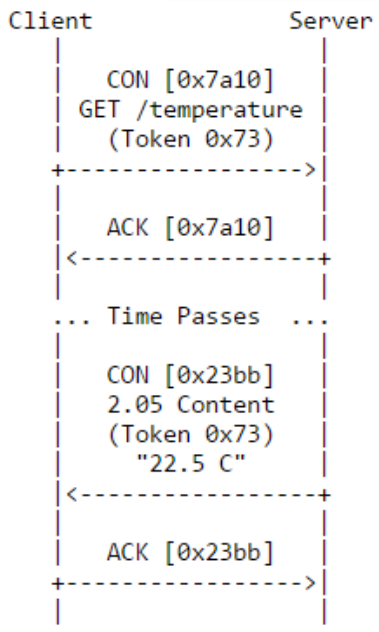


3.5 Dos respuestas piggybacked, una exitosa y otra no encontrada

Los request y las respuestas en CoAP incluyen un Method Code o un Response Code en el propio mensaje. El Token es un elemento opcional utilizado para unir request con respuestas, siendo este diferente del Message ID.

Si la respuesta al request está disponible inmediatamente, ésta se incluye en el propio ACK del mensaje. Esto se denomina respuesta piggybacked (Figura 3.5).

Si por el contrario el servidor no puede responder al instante, el servidor envía un ACK para confirmar que ha recibido la petición y envía la respuesta un tiempo después en un mensaje CON, que tendrá que ser ACK por el cliente. Esto se denomina respuesta separada (Figura 3.6).

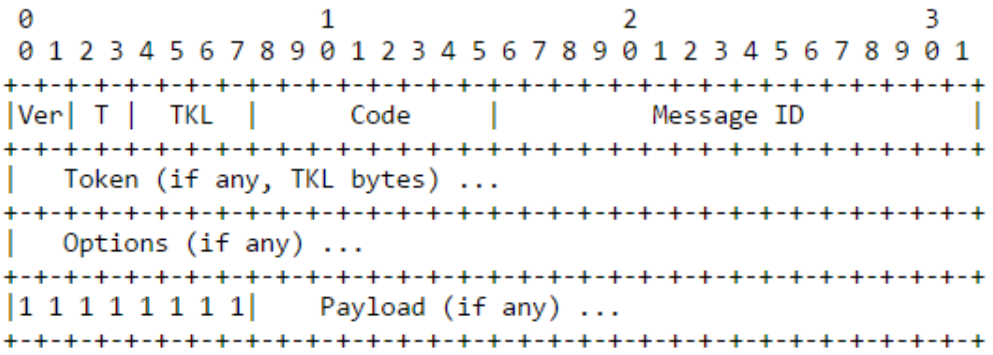


3.6 CoAP respuesta separada a request CON

Formato del mensaje

Como se ha comentado, CoAP se basa en enviar mensajes compactos principalmente a través de UDP, aunque también puede soportar otros protocolos de transporte como SMS, TCP o SCTP.

Sus mensajes se codifican en un formato binario. El mensaje comienza con un encabezamiento de 4 bytes seguido de un Token de longitud variable (entre 0 y 8 bytes), después viene una secuencia opcional de Opciones en formato TLV y finalmente se incluye el contenido del mensaje (Payload).



3.7 Formato del mensaje

El encabezamiento consta de los siguientes campos:

Tabla 4 Encabezamiento mensajes CoAP

Campo	Descripción
Ver	Versión. Número entero de 2 bits. A día de hoy solo hay la 01
T	Tipo. Número entero de 2 bits. Indica si el mensaje es CON (0), NON (1), ACK (2) o Reset (3)
TKL	Longitud del Token. Número entero de 4 bits, de 0 a 8 bytes.
Code	Número entero de 8 bits. Los diferentes códigos se explican más adelante.
Message ID	Número entero de 16 bits.

El **Token** es utilizado para emparejar una respuesta con un request. Podría haberse llamado "request ID". El valor es una secuencia de 0 a 8 bytes y su principal utilidad es le diferenciación entre request concurrentes. El Token podría estar vacío cuando no hay otros Tokens siendo usados y los request están hechos en serie y su respuesta vuelve en forma de piggyback.

Tanto los request como las respuestas pueden incluir una serie de **opciones** que indican una serie de datos sobre el mensaje. CoAP define una serie de opciones que son las mismas tanto para request como para respuestas:

Tabla 5 Opciones para CoAP

No.	Name	Format	Length
1	If-Match	opaque	0-8
3	Uri-Host	string	1-255
4	ETag	opaque	1-8
5	If-None-Match	empty	0
7	Uri-Port	uint	0-2
8	Location-Path	string	0-255
11	Uri-Path	string	0-255
12	Content-Format	uint	0-2
14	Max-Age	uint	0-4
15	Uri-Query	string	0-255
17	Accept	uint	0-2
20	Location-Query	string	0-255
35	Proxy-Uri	string	1-1034
39	Proxy-Scheme	string	1-255
60	Size1	uint	0-4

Las funciones **Uri-Host, Uri-Port, Uri-Path, y Uri-Query** se usan para especificar la fuente de destino de un request a un servidor de origen.

Las funciones **Proxy-Uri y Proxy -Scheme** sirven para hacer un request a un proxy para que éste lo pase o devuelva una respuesta cache del request.

Content format indica el formato para representar el payload y **Accept** para que el cliente pueda especificar que formatos son compatibles con él.

Max-Age indica el tiempo máximo que esa respuesta puede ser guardada hasta que no se considere actualizada.

ETag es una variable para diferenciar entre diferentes representaciones de la misma variable que varíen con el tiempo.

Location-Path y **Location-Query** indican, con una URI relativa, la localización a la que se accede.

If-Match y If-non-Match son dos opciones condicionales que permiten al cliente pedirle a un servidor que realice una acción solo si se cumplen las condiciones indicadas.

Por último, **Size1** proporciona información sobre la representación de un request.

Métodos y Códigos de respuesta

A continuación, se describen los diferentes métodos soportados por CoAP. Esta son los diferentes tipos de request que un cliente puede solicitar:

- **GET:** Este método obtiene la información contenida en la variable identificada por la URI. Si ha sido exitoso se devolverá un código de respuesta 2.05 (content) o 2.03 (valid). Este método es idempotente, su repetición obtiene el mismo resultado.
- **POST:** Este método pide que el contenido del request sea colocado en un nuevo elemento de la variable. Aunque este efecto puede variar en función de la programación del servidor. Si ha sido exitoso devolverá un código 2.01 (created) o 2.04 (changed). Esta función no es idempotente.
- **PUT:** Este método solicita que el contenido del request actualice el contenido de la variable existente. Si ha sido exitoso devolverá un código 2.04 (changed) o 2.01 (created). Esta función es idempotente.
- DELETE: Este método solicita eliminar el contenido de la variable. Si ha sido exitoso devolverá el código 2.02 (deleted). Esta función es idempotente.

Una vez hecho un request se obtiene una respuesta con un Código de respuesta. Este código es un número entero de 8 bits dividido en una clase de 3 bits (los 3 bits más significativos) y un detalle con los 5 bits restantes. El formato es c.dd. Casi todos estos códigos de respuesta son homólogos a los de HTTP con formato cdd.

A continuación, se detallan las diferentes clases y sus detalles correspondientes:

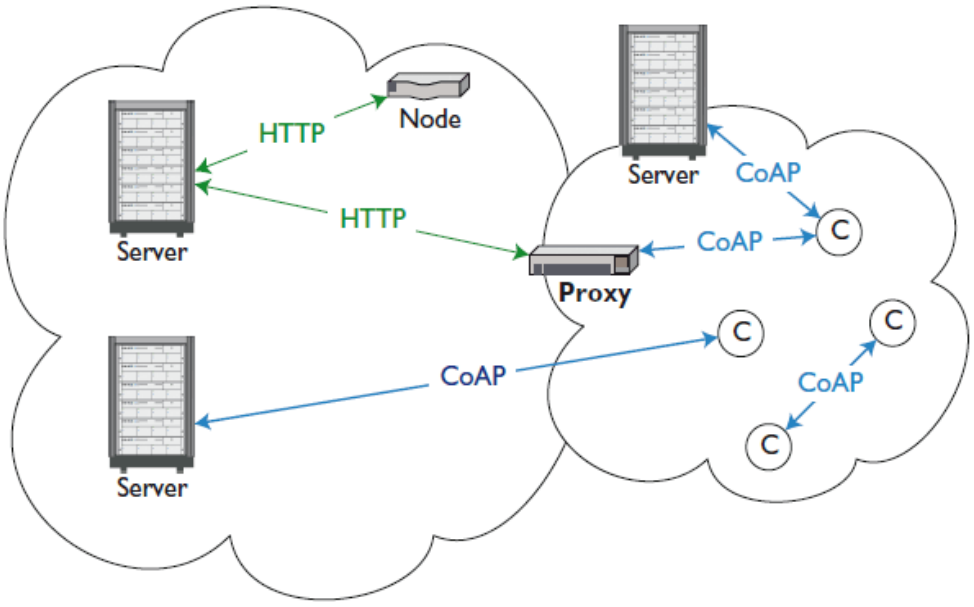
- **2.xx Success**: Éxito, esta clase en un código de respuesta indica que el request ha sido recibido, entendido y aceptado. Puede tener los siguientes detalles:
  - **2.01 Created**: Se ha creado un nuevo elemento en el URI especificado.
  - **2.02 Deleted**: El elemento ha dejado de existir.
  - **2.03 Valid**: Indica que la respuesta identificada por la opción ETag es válida.
  - **2.04 Changed:** Indica que el elemento ha sido modificado.
  - **2.05 Content:** Sirve como OK para función GET e incluye el contenido de la variable pedida.

- **4.xx Client Error**: Indica que el cliente parece haber tenido un error. Puede tener los siguientes detalles:
  - **4.00 Bad Request**: Request mal efectuado.
  - **4.01 Unauthorized**: El cliente no puede efectuar ese método en esa variable.
  - **4.02 Bad Option**: No se pudo entender las opciones enviadas.
  - **4.03 Forbidden**: Prohibido.
  - **4.04 Not Found:** Como el HTTP 404, no se encontró.
  - **4.05 Method Not Allowed:** No se permite ese método.
  - **4.06 Not Acceptable:** No se aceptó ese método.
  - **4.12 Precondition Failed:** Una condición no se cumplió.
  - **4.13 Request Entity Too Large:** El tamaño del request es muy grande para el servidor.
  - **4.15 Unsupported Content-Format:** No se entiende el formato.
- **5.xx Server Error**: Indica que el servidor es consciente de haber tenido un error y no puede efectuar el request. Puede tener los siguientes detalles:
  - **5.00 Internal Server Error**: Error interno del servidor.
  - **5.01 Not Implemented**: No se ha implementado el request.
  - **5.02 Bad Gateway**
  - **5.03 Service Unavailable:** Servicio no disponible en este momento.
  - **5.04 Gateway Timeout:** Se ha producido un timeout.
  - **5.05 Proxying Not Supported:** No se permite proxy.

Cache y Proxy

Los diferentes componentes de una red CoAP pueden cachear respuestas para reducir el tiempo de respuesta y el consumo de ancho de banda para futuros request equivalentes. Para un request concreto, un elemento CoAP no debe usar una versión cacheada a no ser que el método del request coincida con el utilizado para recuperar la respuesta cache, todas las opciones entre el request actual y el guardado coincidan y la respuesta guardad este actualizada. Se considera actualizada si ha transcurrido menos del Max-Age desde que se obtuvo esa copia almacenada.

Los Proxy son componentes de CoAP que pueden realizar request en nombre de otros clientes. Esto puede ser útil para obtener un cache, reducir el tiempo de respuesta o el consumo de energía. A su vez los proxys pueden servir de "traductores" de protocolos. En el caso de CoAP es muy habitual que existan proxys que conviertan un request a HTTP debido a la gran similitud con este protocolo, facilitando así el acceso a un mayor número de componentes de la red.



3.8 Proxy CoAP HTTP

URIs

CoAP utiliza los esquemas *coap* y *coaps* para identificar la localizar los diferentes elementos CoAP. Estos elementos están organizados de forma jerárquica de forma muy similar a los de HTTP. La estructura utilizada es:

"coap:" "//" host [":" port] path-abempty ["?" query]

El host puede ser directamente una dirección IP o puede ser un nombre registrado y se necesitará de DNS para encontrar la dirección de dicho host. Un ejemplo de URI de CoAP podría ser: *coap://example.com:5683/~lights/data.xml*

Seguridad

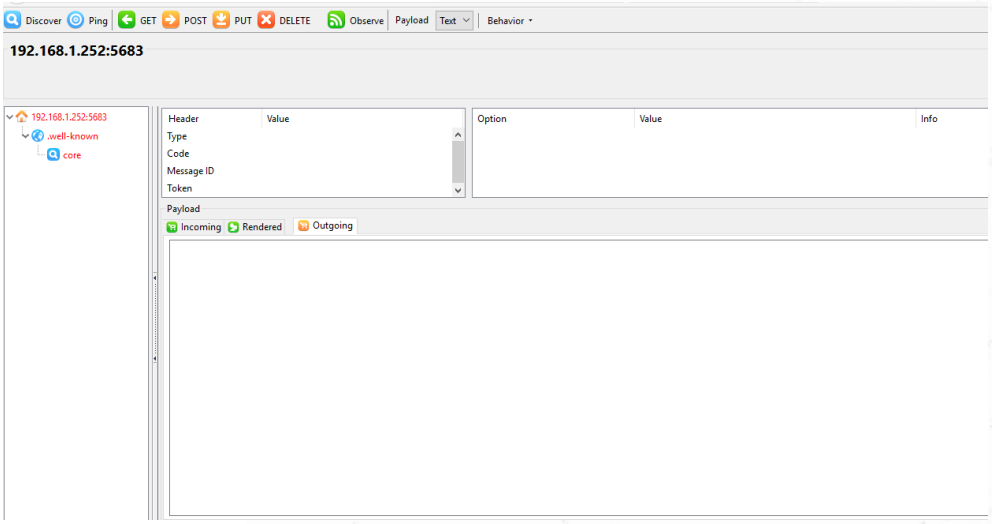
CoAP está protegido a través de DTLS, protocolo específico para UDP. Para realizar una transmisión de datos entre cliente y servidor utilizando CoAP de forma segura, se necesita, antes de poder enviar el primer request, realizar un DTLS "handshake" entre cliente y servidor para poder establecer una sesión.

Para emparejar los ACKs con los CONs, aparte de tener el mismo Message ID, ambos deben haber sido enviados en la misma sesión DTLS. Las conexiones entre componentes suelen mantenerse abiertas al acabar un intercambio de mensajes para futuros intercambios. Éstas son cerradas solo si se requiere recuperar los recursos utilizados por dicha conexión. Cerrarlas al acabar cada intercambio de mensajes sería muy ineficiente.

3.3 IMPLEMENTACIÓN EN EL LABORATORIO

El laboratorio de Automatización donde se realizaron las pruebas con el SNMP no dispone de software ni hardware compatible con CoAP por lo que las pruebas realizadas con este protocolo no pudieron ser realizadas sobre los mismos componentes ni con el mismo software TIA Portal con el que se implementaron las de SNMP. No obstante, las funciones y tipo de pruebas han intentado ser lo más similares posibles a las de SNMP para poder determinar si ambos pueden desempeñar las mismas funciones.

Como se detalla anteriormente, para las pruebas de CoAP se utilizó una herramienta bastante útil, el Add-On de Mozilla Firefox llamado Copper (Cu) que puede ser descargado aquí. Está extensión del navegador Mozilla Firefox fue desarrollada por Matthias Kovatsch un investigador especializado en Internet of Things de la universidad ETH Zúrich. Este Add-On del navegador, aparte de ser de gratuita instalación, convierte la pantalla del navegador Mozilla Firefox en una interfaz bastante intuitiva y fácil de manejar con la cual el usuario puede desempeñar las funciones de Cliente interactuando con un servidor que responda a CoAP.

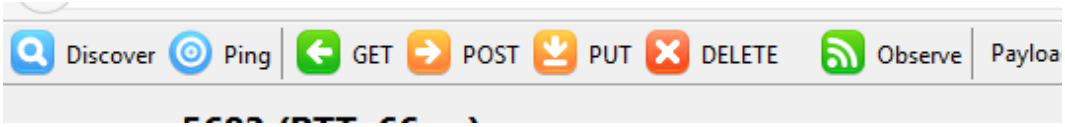


3.9 Interfaz Copper para Firefox

La interfaz aparece en el navegador una vez se introduce en la barra de navegación de éste una URI que sea identificada como CoAP (coap://…).

Para la prueba utilizaremos dos servidores disponibles online específicamente para probar CoAP, el primero y principal coap://vs0.inf.ethz.ch:5683 diseñado por la propia ETH Zurich para la prueba de Copper, el segundo coap://coap.me:5683. (El numero 5683 simplemente representa el número de puerto UDP preestablecido para CoAP)

El primer componente de la interfaz, y el más útil es la barra de herramientas en la parte superior, aquí se encuentran todas las funciones disponibles en CoAP para poder interactuar con el servidor.



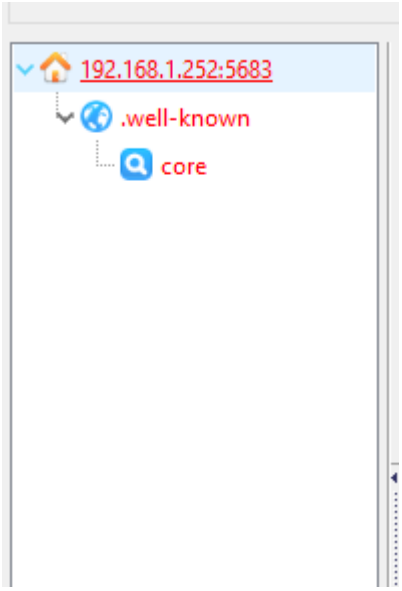
3.10 barra de herramientas Copper

Lo primero que se debe hacer es realizar un Ping al servidor que se desea controlar, de esta manera se sabrá si el servidor está Online, si soporta CoAP, y cuál es el RTT (Round-Trip Time) para saber si todo está en orden.

3.11 Respuesta a Ping para confirmar lo correcto del servidor

Después tenemos el árbol de elementos, inicialmente siempre enseñará el nombre del servidor y su extensión /.well-known/core. Para obtener el resto de elementos, es necesario pulsar el botón Discover de la barra de herramientas, desplegando el árbol de elementos completo como se puede observar en la figura 3.13.

Una vez despeglado el arbol, se pueden observar todas los elementos contenidas en el servidor. Estos elementos funcionan con el mismo concepto que las varaibles de SNMP, cada uno tiene unos características, puede ser solo leido, leido y modificado, soporta Observe, aporta un contador, un numero, un texto… etc. La gran diferencia con las variables SNMP es que estos no están reguladas ni indexadas por un convenio como los OID, cada uno depende de como haya sido programado el servidor y del programador en cuestión depende el nombre, las funciones y accesos que le hayan sido otorgadas a cada uno.

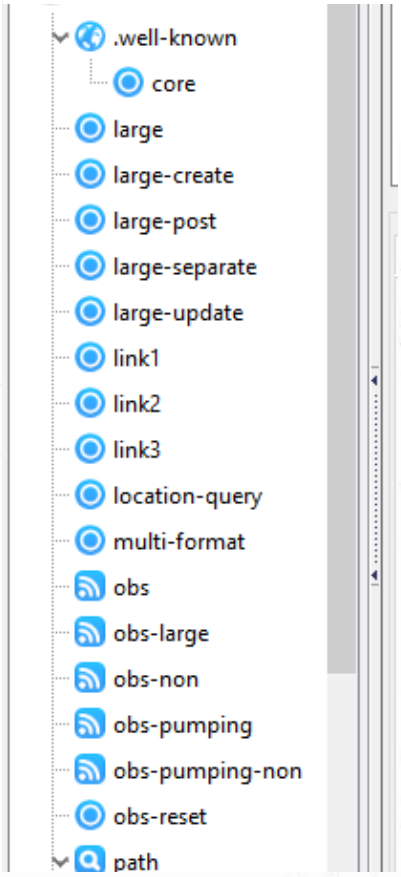


3.12 árbol de variables inicial

Por ello, es habitual que en el propio core, el programador incluya algunas características y definición de los elementos, sobre todo si este servidor pretende ser usado por gente ajena al diseñador, como es el caso que nos atañe.

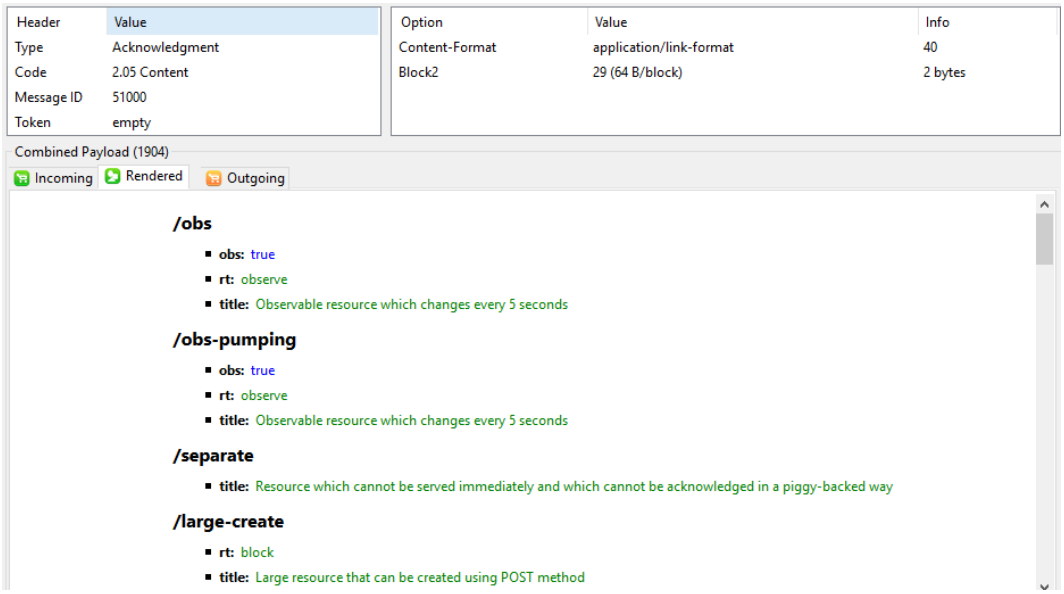
Otros aspectos a tener en cuenta del interfaz son la pestaña de Behavior en la barra de herramientas (figura 3.15), dónde se puede editar las características de las funciones, si se queire ACK o no, el tamaño de los bloques de información etc.

Por último, el resto de la interfaz se reparte entre la parte inferior de la pantalla en la que se leen los resultados recibidos, tanto en plain text (pestaña Incoming) como Renderizados (pestaña Rendered) si requieren de ello, y donde se escriben los datos a enviar (pestaña Outgoing) y la parte superior de donde se plasman los datos del último



3.13 árbol de variables tras Discovey

mensaje recibido al cliente, desde ID del demensaje, tamaño del mensaje, tipo de mensaje, formato, Token etc. Esta última herramienta es muy útil para poder interpretar lo que ocurre con el protocolo en cada caso.



3.14 Copper, pantalla de resultados con características del mensaje.

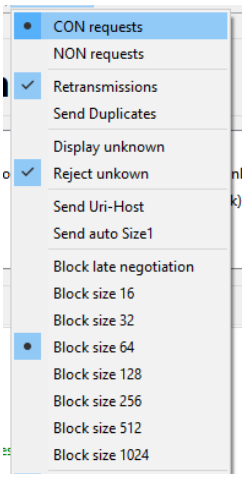
En esta última figura, además de las dos pantallas explicadas anteriormente, se puede observar lo mencionado en la página anterior, el resultado de realizar la operación GET en el core: una pequeña explicación de las diferentes variables, indicando si soportan la función Observe y que se encuentra en cada una de ellas etc. Resultado que además se puede apreciar renderizado en la pestaña Rendered y no en el texto crudo de la pestaña Incoming.

Una vez explicado la herramienta Copper y sus componentes, probado la conexión mediante Ping y desglosado el árbol de elementos mediante Discover, se procede a probar las diferentes funciones CoAP.

GET

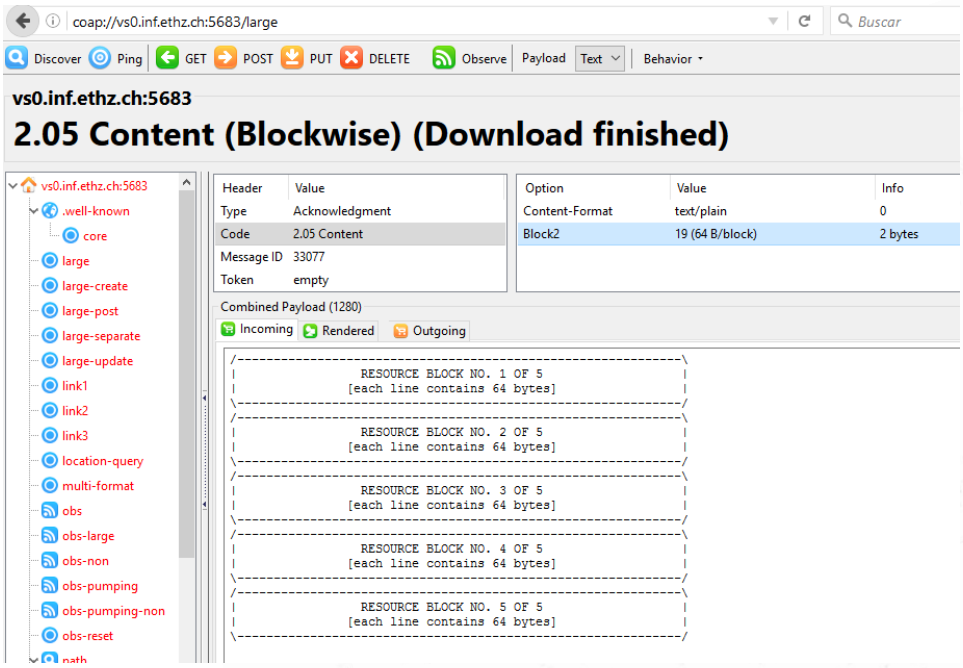
La función Get tiene exactamente la misma utilidad que la de SNMP, de un elemento en concreto del servidor, obtener su contenido. Para ello simplemente se ha de seleccionar el elemento del árbol de elemento, ajustar las características del request en el despegable Behavior y presionar la función GET.

En este caso se elige un request Confirmable (CON) esperando un Acknowledgment y un Block Size de 64 Bytes (figura 3.15)



3.15 Behavior Get

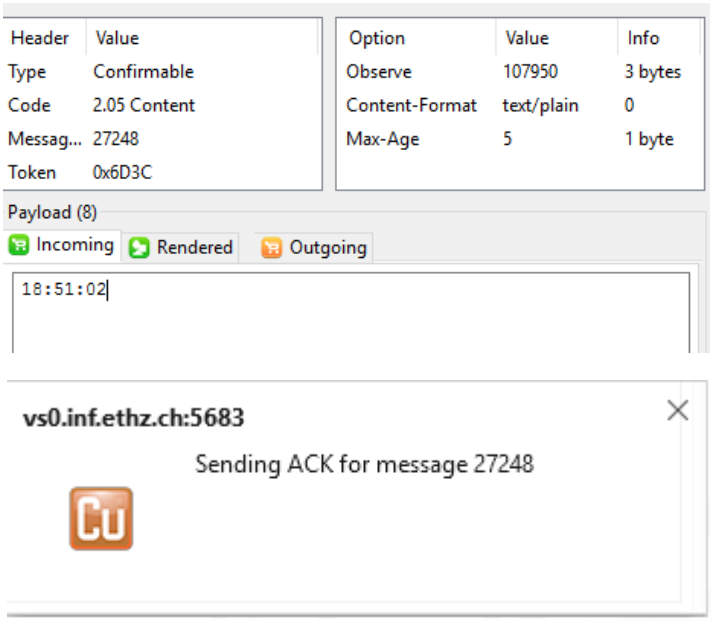
En la figura 3.16 se puede observar el resultado obtenido. El resultado descargado del elemento son una serie de bloques de texto que se representan en la pantalla incoming. La parte más interesante son las características del mensaje recibido que se puede observar en las pantallas superiores. Se ve claramente que se trata de un mensaje Acknowledgment, como se espera de un request CON. Se recibe también un Response Code de 2.05 (content) como también se puede esperar de un request GET. Se aprecia el número de bloques (19) de 64 bytes cada uno. Por último, destacar el Message ID, para que el agente sepa a que request corresponde esta respuesta recibida y el Token, que, en este caso, al no haber ningún otro Token en uso y recibir la respuesta en el propio mensaje de Acknowledgment (piggyback response), el cliente no necesita un Token para poder ubicarla.



3.16 Respuesta Get CoAP

OBSERVE

La siguiente función a probar es la función Observe, que tiene un comportamiento muy similar al Trap de SNMP. Esta función lo que hace es crea una "subscripción" a un elemento, con un tiempo determinado y el servidor responde enviando un mensaje con el valor de ese elemento de manera periódica al cliente.

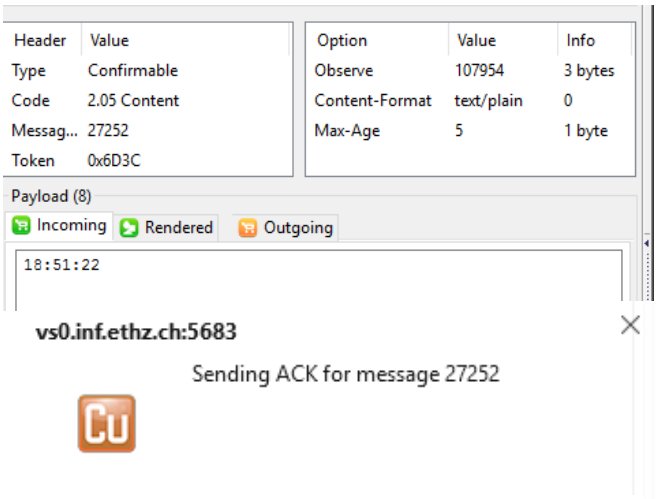


3.17 CoAP Observe 1er mensaje

En este caso se realiza un Observe de un elemento que devuelve la hora de ese momento. El observe lo que hace es que el servidor envía mensajes al cliente por iniciativa propia. Por ello, en este caso se puede observar como hay diferencias con las características del mensaje obtenidas en el GET. Para empezar, el tipo del mensaje, pasa de Ack a Confirmable, en este caso es el servidor el que pide la confirmación al cliente de que ha recibido el mensaje, no viceversa, de ahí que el tipo sea Confirmable. A cambio, para satisfacer la petición del servidor, como se puede ver en la parte inferior de la figura 3.17, es el cliente (Copper en este caso) el que le envía un Ack al servidor para confirmar que ha recibido el mensaje, coincidiendo el Message ID para que sepa el servidor que mensaje ha sido recibido. Se ve después que el Code es 2.05 de nuevo, probando que el contenido (la hora) ha sido entregado correctamente.

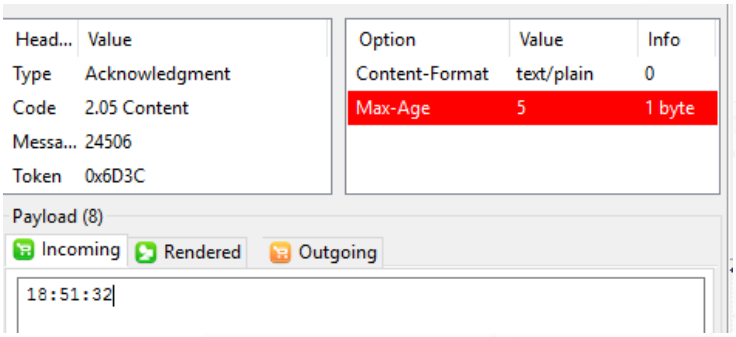
Por último, destacar dos novedades respecto a lo visto en GET. En primer lugar, el Token, en este caso sí que es importante ya que se enviarán multitud de mensajes. Mediante el Token el cliente sabe que todos los que lo lleven responden al mismo Observe. En segundo lugar, el Max-Age, este dato corresponde a cada cuanto

tiempo se envía un nuevo mensaje al cliente, por tanto, ese dato representa la antigüedad máxima que puede tener ese valor obtenido del elemento.



3.18 CoAP Observe 2º mensaje

Como se puede observar en el segundo instante obtenido del Observe (figura 3.18) el Token es el mismo que en el primer mensaje, para que el cliente sepa que pertenece a otro mensaje del mismo Observe. Por el contrario, este mensaje tiene un Message ID diferente, al ser otro mensaje, que coincide de nuevo con el ACK enviado por Copper para confirmar su recepción. Tiene también otro valor ya que han pasado 20 segundos, y se puede obsrvar también en la variable Observe que funciona como contador de mensajes enviados, ha aumentado en 4, queriendo decir que se han enviado 4 mensajes entre la primera y segunda captura.



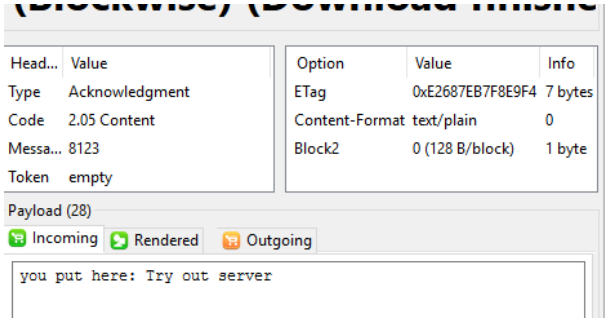
3.19 CoAP

Esta última imagen (figura 3.19) corresponde con la cancelación del Observe y el último mensaje recibido. Como se espera se puede seguir viendo el mismo Token, para saber que se refiere a la misma ristra de mensajes originados por el mismo Observe. Tiene de nuevo otro Message ID, demostrando que es otro mensaje, pero tiene un par de peculiaridades con respecto a los anteriores. En primer lugar, el tipo de mensaje cambia de Confirmable a Ack, ya que en este caso, este último mensaje enviado por el servidor, es la confirmación (ACK) de haber recibido la solicitud de cancelación al Observe enviada por el cliente. Este Ack ejerce de

Piggyback Message y por tanto no solo confirma sino que trae el úlitmo bloque de información. La otra diferencia es el Max-Age en rojo, esto quiere indicar que han pasado ya más de 5 segundos desde la cancelación, y que, por tanto, ese dato recibido (la hora) tiene una antigüedad mayor que la estipulada por ese dato, por lo que no es fiable.

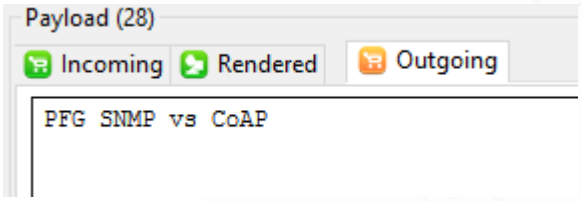
POST

Tras haber probado las dos funciones de obtención de datos, se va a probar ahora las funciones que permiten la modificación de los elementos. A diferencia de SNMP que solo existía una única función GET, aquí se dispone de hasta 3, POST, PUT y DELETE. Empezaremos probando la función POST.



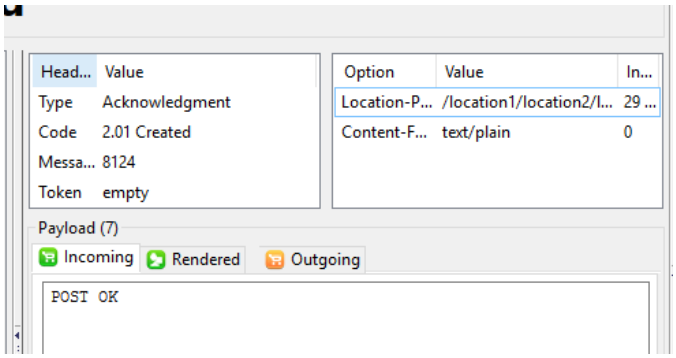
3.20 CoAP Get de variable a modificar

Como se explicó en la teoría, la acción realizada exacta con la función POST es variable según se haya programado en el servidor. Inicialmente se realiza un GET para saber que se encuentra en este elemento concreto, se ve que contiene el texto “you put here: Try out server” (Figura 3.20).



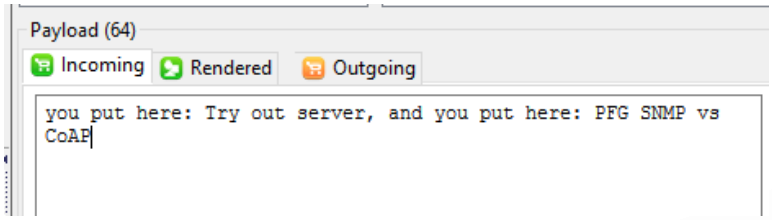
3.21 contenido POST

Una vez obtenido el estado incial se escribe en la pestaña Outgoing un texto, en este caso el texto “PFG SNMP vs CoAP” (figura 3.21) y se realiza la función POST.



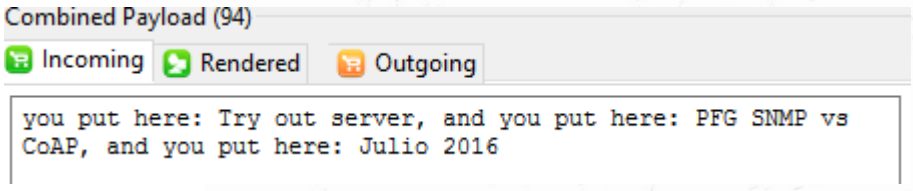
3.22 resultado tras POST

En el resultado de la función POST, se ve que el Code en este caso es 2.01, indicando que se ha creado una entidad nueva, y es un mensaje Piggybacked en un ACK ya que la petición inicial de POST se pidó como Confirmable. El ACK a su vez incluye un mensaje de texto incluido por el programador para la pantalla "POST OK" indicando que se ha efectuado la función correctamente. También se puede observar que devuelve un Location-Path indicando donde se guarda los nuevos datos, y el tipo de contenido que se ha aportado "plain/text" en este caso.



3.23 GET tras POST

Si ahora se realiza un GET para saber el estado del elemento, se puede observar que la función POST, en este caso, como ha sido programada en el servidor y suele ser su uso habitual, añade un elemento nuevo al elemento con los datos enviados, con el mensaje previo "and you put here" sin eliminar lo existente previamente. Esto se podía suponer debido al Code del mensaje 2.01 (create), explicando que crea un elemente nuevo, en vez de 2.04 (Change) que indicaría que se cambia lo existente en el elemento.

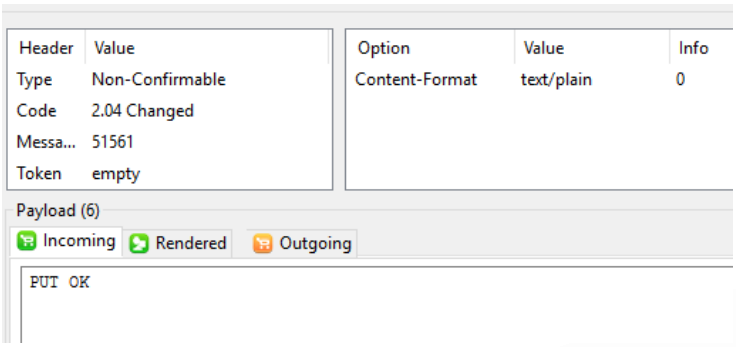


3.24 Get Tras dos POST

Se confirma la suposición de la programación del servidor al añadir otro POST con el texto "Julio 2016" y ver el resultado de la figura 3.24, mantiene lo existente y añade otro elemento nuevo.

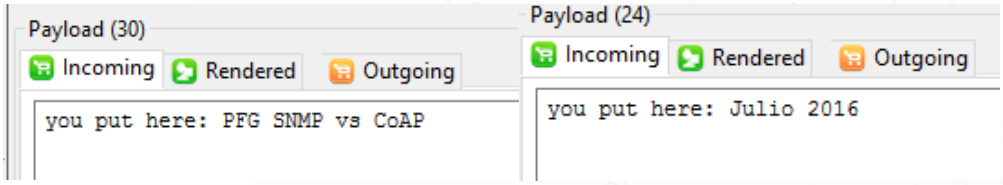
PUT

La función PUT tiene un comportamiento similar al de POST, sirve para poder modificar los elementos existentes en el servidor. Partiendo del mismo elemento del POST que incluía el mensaje "you put here: Try out server" (figura 3.20) se realiza una función PUT con el mismo mensaje que el de la función POST "PFG SNMP vs CoAP" (figura 3.21), en este caso, para variar un poco el resultado, en la pestaña Behavior se indicará que sea un NON (non-confirmable) request.



3.25 Resultado PUT

Se obtiene un mensaje parecido al del POST, con algunos detalles distintos, el código es en este caso 2.04 (changed) en vez de 2.01, dando a pensar que la trasformación ha sido diferente a la del POST y el tipo cambia a Non-Confirmable al haber enviado un request como tal, no requiriendo por tanto un Ack. Se obtiene también el mensaje introducido por el programador PUT OK confirmando la correcta ejecución de la función.



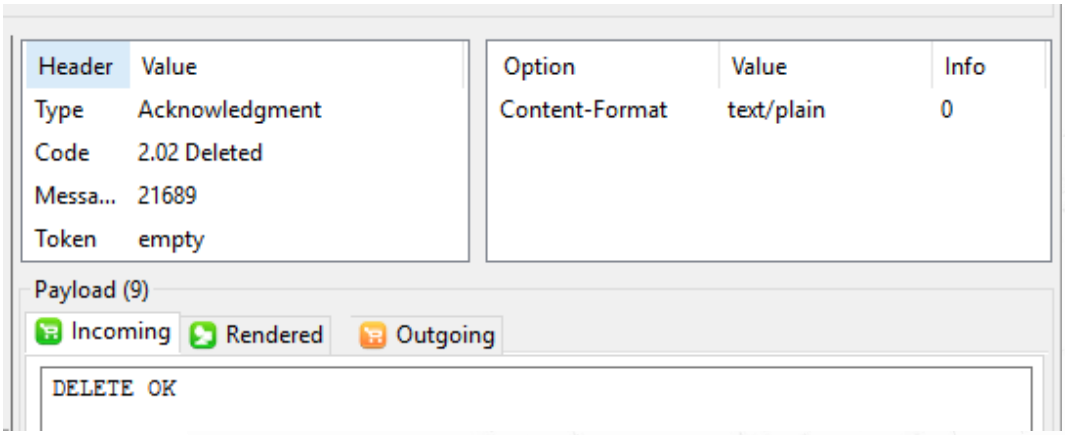
3.26 GET tras un PUT y tras 2 PUT

Si se realiza el GET posterior para ver el resultado del elemento (figura 3.26), se puede apreciar que el PUT en este sustituye el elemento inicial por el insertado con la función. Esto se podía intuir ya que el código del mensaje resultante era 2.04 (changed) y no 2.01 (created) como en el caso del POST. Para confirmar la suposición se realiza, al igual que en el POST, otro PUT con el texto "Julio 2016" y se aprecia que el resultado, en efecto, sustituye al elemento existen por el introducido (Figura 3.26).

DELETE

Por último, la función que se probará es la DELETE, en este caso, como su nombre indica, modifica el contenido del elemento eliminándolo.

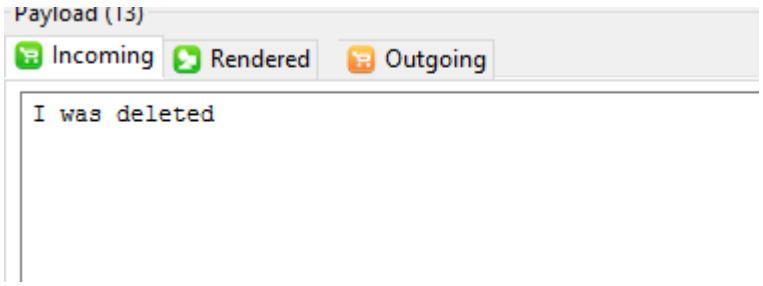
Partiendo de nuevo del elemento utilizado para probar POST y PUT con valor inicial "you put here: Try out server" (figura 3.20) se ejecuta la función DELETE obteniendo el siguiente resultado:



3.27 Resultado DELETE

Se puede apreciar que el Código es 2.02 (Deleted) indicando que se ha eliminado el contenido existente, y el tipo de mensaje es ACK ya que se pidió un Request CON.

Si realizamos un GET al elemento obtenemos el siguiente mensaje:



3.28 Get tras DELETE

En realidad, el elemento debería estar vacío, pero el programador del servidor ha decidido que si se borra el elemento aparezca este mensaje, confirmando la eliminación correcta de su contenido. No obstante, se puede apreciar que es diferente a lo anteriormente visto con PUT y POST ya que este mensaje muestra que no hay ningún elemento, ni el original ni uno nuevo, ya que éstos siempre empezaban con las palabras “you put here:”.

Con el DELETE concluye la comprobación práctica de las diferentes funciones de CoAP y su comportamentiento a través de Copper (Cu).

3.4 CONCLUSIONES COAP

Tras lo estudiado de forma teórica y práctica del protocolo CoAP se puede describir las siguientes características del protocolo.

CoAP es un protocolo muy reciente con vistas al futuro. Es un protocolo con la categoría de Proposed Standard desde junio de 2014 nada más. Esto hace que aun esté por desarrollar y exprimir de forma más efectiva y probablemente saldrán en un futuro versiones que lo mejoren. No obstante, a pesar de su juventud, es un protocolo fácilmente utilizable y comprensible que encaja muy bien en el Internet de hoy en día. Su fácil comparación y sus similitudes con HTTP, a parte de su capacidad de proxy con el mismo, hacen que pueda abarcar y alcanzar gran cantidad de los dispositivos conectados a la red global de IoT. Además, s

En cuanto a lo que Network Management se refiere, se ha podido comprobar que CoAP puede realizar todas las funciones necesarias para poder manejar de forma satisfactoria los componentes de una red de comunicación. Las mayores limitaciones que tiene es lo poco normalizado de los servidores y funciones, haciendo difícil la adhesión de elementos externos o la interconexión de redes diferentes ya que podrían llegar a no “entenderse”.

Aun así, si lo que se desea es construir una red de comunicación de cero en la actualidad, por la capacidad de personalización de la misma, la proyección de futuro y las oportunidades que puede ofrecer la conexión al IoT, probablemente CoAP sea el protocolo más adecuado.

Capítulo 4 COMPARATIVA Y CONCLUSIONES

Tras haber estudiado ambos protocolos tanto de manera teórica como práctica, se puede concluir que ambos pueden realizar las funciones del Network Management de manera satisfactoria por lo que ambos podrían ser perfectamente válidos para esa función a pesar de protocolos radicalmente opuestos. La elección de uno u otro viene más determinada la situación concreta en la que se necesite emplear uno de estos protocolos y los conocimientos y preferencias personales.

Por un lado, SNMP es un protocolo clásico, exclusivo de Network Management y es lo que es. Todo en él está normalizado, delimitado, indexado y comprobado. Uno sabe a lo que atiene cuando lo elige y conoce perfectamente lo que le puede aportar. Lleva más de 20 años siendo el protocolo de referencia de Network Management y la mayoría de componentes del mercado específicos para esto lo soportan y lo llevan implementado de serie.

Por otro lado, CoAP es un protocolo muy reciente, con apenas 2 años de vida. El Network Management es una de las aplicaciones que este puede tener, pero no se sabe aún su techo ya que seguirá evolucionando y saldrán versiones nuevas de él. Su similitud con HTTP lo hace sencillo de comprender e implementar. Tiene una gran proyección de futuro permitiendo la conexión al Internet of Things. Su falta de rigidez en sus delimitaciones lo hace mucho más personalizable para los requisitos particulares del usuario que SNMP, aunque esta falta de normalización y estandarización hace que sea más complicado la comunicación entre redes independientes o la implementación de componentes externos a una red.

Por todo esto, la elección de uno u otro depende principalmente del uso que se le vaya a dar y la preferencia personal. Si lo que se pretende es simplemente controla una red ya existente y nada más, SNMP probablemente sea el protocolo más pragmático para ello, ya que la red probablemente lo soporte. Si, por el contrario, uno quiere montar una red de cero, con proyección de futuro, conexión a IoT y con mayor personalización del control y de las funciones que ésta realiza, lo más

adecuado sería la elección de CoAP. Resumiendo, SNMP es el protocolo más clásico, seguro y pragmático para este uso, pero CoAP ofrece muchas más posibilidades y proyección de futuro.

BIBLIOGRAFÍA

[1] Mauro, Douglas R. & Schmidt, Kevin J., (2001), *Essential SNMP*, Sebastopol, CA, USA: O'Reilly & Associates.

[2] SNMP: https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

[3] CoAP: https://en.wikipedia.org/wiki/Constrained_Application_Protocol

[4] MIB: https://en.wikipedia.org/wiki/Management_information_base

[5] Base de datos OID: http://www.alvestrand.no/objectid/top.html

[6] Conversor hexadecimal: http://www.unit-conversion.info/texttools/hexadecimal/

[7] OID: https://en.wikipedia.org/wiki/Object_identifier

[8] Character Encoding: http://support.sas.com/documentation/cdl/en/spdsug/64018/HTML/default/viewer.htm#n12oqmbf5ny76qn1d4ycnfdc8vzh.htm

[9] Copper: http://www.vs.inf.ethz.ch/publ/papers/mkovatsc-2011-dcoss-copper.pdf

[10] Copper: http://reference.bitreactive.com/tutorials/add-coap-to-your-application.html

[11] ASN.1: https://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One

[12] CBC: https://es.wikipedia.org/wiki/Reglas_de_codificaci%C3%B3n_b%C3%A1sicas

[13] IoT https://en.wikipedia.org/wiki/Internet_of_things

[14] 6LoWPAN: https://en.wikipedia.org/wiki/6LoWPAN

[15] RFC SNMPv1: https://tools.ietf.org/html/rfc1157

[16] RFC SNMPv3: https://tools.ietf.org/html/rfc3411

[17] RFC SNMPv3: https://tools.ietf.org/html/rfc3412

[18] RFC SNMPv3: https://tools.ietf.org/html/rfc3413

[19] RFC SNMPv3: https://tools.ietf.org/html/rfc3414

[20] RFC SNMPv3: https://tools.ietf.org/html/rfc3415

[21] RFC SNMPv3: https://tools.ietf.org/html/rfc3416

[22] RFC SNMPv3: https://tools.ietf.org/html/rfc3417

[23] RFC SNMPv3: https://tools.ietf.org/html/rfc3418

[24] RFC CoAP: https://tools.ietf.org/html/rfc7252

[25] DTLS: https://es.wikipedia.org/wiki/Datagram_Transport_Layer_Security

[26] Protocolos de red: https://en.wikipedia.org/wiki/List_of_web_service_protocols

[27] RFC HTTP: https://tools.ietf.org/html/rfc7230

Parte II ANEXO

Capítulo 1 SIGLAS Y ACRÓNIMOS

#

6LoWPAN: IPv6 Over Low power Wireless Personal Area Networks

A

ASN-1: Abstract Syntax Notation One (notación sintáctica abstracta 1)

ACK: Acknowledgment

B

BER: Basic Encoding Rules (reglas de codificación básicas)

C

CoAP: Constrained Application Protocol (protocolo de aplicación restringida)

CMIP: Common Management Information Protocol (protocolo de administración de información común)

CMOT: CMIP over TCP/IP

CBC: Cipher Block Chaining (encadenamiento de bloques cifrados)

CPU: Central Processing Unit (unidad central de procesamiento)

CON: Confirmable request

D

DES: Data Encryption Standard

DTLS: Datagram Transport Layer Security protocol

E

ETH Zurich: Eidgenössische Technische Hochschule Zürich (Escuela Politécnica Federal de Zúrich)

EGP: Exterior Gateway Protocol

EPC: Electronic Product Code (código electrónico de producto)

G

GSDML: General Station Device Markup Language

H

HTTP: Hypertext Transfer Protocol (protocolo de transferencia de hipertexto)

HMAC: Hash-based message Authentication Code (código de autentificación de mensajes en clave hash)

I

IP: Internet Protocol (Protocolo de Internet)

IETF: Internet Engineering Task Force

IAB: Internet Architecture Board

ISO: International Organization for Standardization (Organización Internacional de Normalización)

ICMP: Internet Control Message Protocol (protocolo de mensajes de control de Internet)

IoT: Interne of Things

IoT-GSI: Global Standards Initiative on Internet of Things

IoTSF: Internet of Things Security Foundation

M

M2M: Machine to Machine (máquina a máquina)

MIB: Management Information Base (Base de Información Gestionada)

MD5: Message-Digest Algorithm 5 (Algoritmo de Resumen del Mensaje 5)

MIT: Massachusetts Institute of Technology (Instituto Tecnológico de Massachusetts)

MQTT: Message Queuing Telemetry Transport

N

NMS: Network Management System (sistema administrador de red)

O

OMA LWM2M: Open Mobile Alliance Lightweight Machine to Machine protocol

OSI: Open System Interconnection (interconexión de sistemas abiertos)

OID: Object Identifier (identificador de objeto)

P

PDU: Protocol Data Unit (unidad de datos de protocolo)

PLC: Programmable Logic Controller (Controlador lógico programable)

PROFINET: Process Field Network

R

RFC: Request for Comment

ROM: Read-Only Memory (memoria de solo lectura)

RAM: Random Access Memory (memoria de acceso aleatorio)

S

SNMP: Simple Network Management Protocol (protocolo simple de administración de red)

SGMP: Simple Gateway Monitoring Protocol

SHA: Secure Hash Algorithm (Algoritmo de Hash Seguro)

SMI: Structure of managed information (estructura de la información gestionada)

SIMATIC: Siemens Automatic

SMS: Short Message Service (servicio de mensajes cortos)

SCTP: Stream Control Transmission Protocol

T

TCP: Transmission Control Protocol (protocolo de control de transmisión)

TIA Portal: Totally Integrated Automation Portal (automatización totalmente integrada)

TLV: Type-Length-Value (tipo-longitud-valor)

U

UDP: User Datagram Protocol

URI: Uniform Resource Identifier (identificador de recursos uniforme)

USM: User-based Security Model (modelo de seguridad basado en el usuario)