Resource-constrained Industrial Things - Proposal for the Adaptation of CoAP to EtherNet/IP™

Jonas Green
Development engineer
Björn Otterdahl
Team Manager CIP Technology
HMS Industrial Networks

Presented at the ODVA
2017 Industry Conference & 18th Annual Meeting
February 21-23, 2017
Palm Harbor, Florida, USA

Abstract

The idea of the Internet of Things to act and sense in the physical world have opened up for new innovations within the communication technology area. A resource constrained device such as a sensor can hardly handle the complexity of a transport protocol like TCP to produce a simple sensing value when it also must consider limited resources in terms of computation power and energy. There are multiple communication protocols under development targeting Internet of Things devices that have achieved reliable communication with minimal resource usage both on the device and on the network. Inspired by previous work in extending EtherNet/IP™ to resource-constrained industrial things this paper investigates the Constrained Application Protocol (CoAP) and the possibility to adapt it to the CIP™ protocol. CoAP is a transport protocol built upon the IP and the UDP protocol designed for use with constrained devices and constrained networks. Previous work put the responsibility to handle the lost reliability on the application when replacing the TCP protocol with UDP. Investigations shows that this is not necessary. For resource constrained devices, handling small amounts of data, CoAP together with UDP replaces the TCP protocol with only a 4-byte fixed header and a compact encoding of options for additional information without compromising the reliability of the network. CoAP supports confirmable and non-confirmable messages, has a basic set of methods to interact with resources of a device and identifies resources by using Uniform Resource Identifiers (URIs). This design conforms well to the CIP protocol which requires handling of explicit messaging and IO connections, possibility to address objects and attributes of the CIP object model and services to interact with those. CoAP together with single pair Ethernet would create a good communication solution for resource constrained industrial things. With the coming transition to IPv6 there will also be several wireless physical layers suitable for resource constrained devices such as the Bluetooth wireless technology, 6LoWPAN and the ZigBee Standard. With a device level network using IP addressing the CIP networks could over time converge towards one common addressing mechanism. To take advantage of the features of CoAP, IP addressing on the device level network and multiple physical layers for constrained devices this paper presents a proposal of how CoAP can be adapted to CIP to handle explicit messaging and IO connections to resource constrained industrial things of a CIP network.

Keywords

CoAP, IPv6, DTLS, Industrial Internet of Things

Definition of terms

CoAP – Constrained Application Protocol
MQTT – MQ Telemetry Transport
DDS – Data Distribution Service
AMQP – Advanced Message Queueing Protocol

1 Introduction

The idea of the Internet of Things to act and sense in the physical world have opened up for new innovations within the communication technology area. A resource constrained device such as a sensor can hardly handle the complexity of a transport protocol like TCP to produce a simple sensing value when it also must consider limited resources in terms of computation power and energy.

On the ODVA 2015 Industry Conference Dayin Xu and Paul Brooks presented a white paper on *"Extending EtherNet/IP™ to Resource-Constrained Industrial Things"* [1] which introduced the idea to run EtherNet/IP solely over the UDP protocol, excluding the TCP protocol. The white paper showed that the processing power and memory consumption needed for such application was significantly reduced using this technique. The white paper also discussed the issues with the lack of reliability in the UDP protocol and it suggested to put the responsibility of building a reliable protocol on the application layer.

This white paper presents one way of solving the reliability issues with the UDP protocol that was addressed in the *"Extending EtherNet/IP™ to Resource-Constrained Industrial Things"* white paper. By introducing the Constrained Application Protocol (CoAP), a newly developed protocol targeting IoT devices, between the UDP layer and the application layer, the reliability problem is solved in an efficient way. The CoAP protocol is a lightweight protocol when it comes to the amount of data added into the frames sent on the network as well as in memory and performance requirements on the host system.

This white paper also builds on and extends the work of the bachelor thesis project *"Resource constrained Industrial IoT device"* [2] by Henrik Wernersson, Yassin Atwa that was made as a collaboration between HMS Industrial Networks AB and Halmstad University in Sweden during the spring of 2016. This thesis project investigated possible options to reduce the complexity of EtherNet/IP using existing IoT protocols. In the thesis project CoAP, MQTT, DDS and AMQP protocols were compared and ranked on complexity and similarities with the EtherNet/IP protocol. The conclusion of the thesis project was that CoAP was the protocol that was best suited for being incorporated with the EtherNet/IP protocol over UDP. The thesis project also incorporated a prototype implementation of CoAP together with the HMS CIP stack.

This white paper gives an introduction to the CoAP protocol, what problem that is solved by using this protocol and finally it presents a way of mapping the addressing scheme of the CoAP protocol to the addressing scheme of EtherNet/IP.

2 Constrained Application Protocol

The Constrained Application Protocol (CoAP) [3] is developed to be a machine-2-machine protocol for constrained devices. To reduce the header overhead and parsing complexity it is built on top of the transport protocol UDP. It implements a messaging protocol with request/response interactions and a lightweight reliability mechanism to handle re-transmissions and duplicate messages detection. It has also support for security with DTLS.

Logically CoAP could be divided into two layers (see Figure 1), a messaging layer dealing with UDP to transport messages asynchronously and a request/response layer handling interactions by using Method and Response codes.

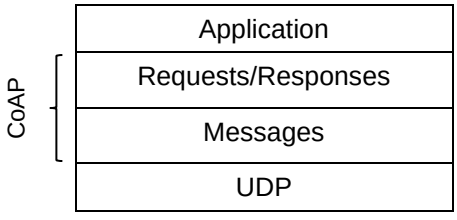

Figure 1: Logical layering of CoAP

CoAP messages are encoded in a simple binary format starting with a fixed 4-byte header. The header is followed by a Token of variable length, any number of options in Type-Length-Value (TLV) format and finally an optional payload. This is visualized in Figure 2.

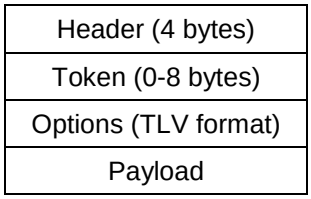

Figure 2: A complete CoAP message

2.1 CoAP header

The header consists of the CoAP version number, message type, token length, message code and a message ID as shown in Figure 3. The current version of CoAP is 1.

Byte	0								1								2								3							
Bit	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Fields	Ver		T		TKL				Code								Message ID															

Figure 3: Illustration of the CoAP header with its parameters

CoAP has four types of messages; Confirmable, Non-confirmable, Acknowledgement and Reset. A Confirmable message requires an Acknowledgement message in return while Non-confirmable does not. The Acknowledgement message could either include eventual response data, known as a piggybacked response, or be empty while the response data is transmitted later in a Confirmable message. A Confirmable message will be re-transmitted if no Acknowledgement message is received in return within a specified timeout. The timeout is then doubled for every re-transmission until the maximum allowed re-transmission count is reached. The Reset message is used when a recipient is not at all able to process a Confirmable or Non-confirmable message.

The message code of the CoAP header is divided into two sections, class and detail. The class consists of three bits and tells if the message is either a request, a success response, a client error response or a server error response. The detail section uses the remaining 5 bits to hold a request method or a response code dependent on what message code class that is set. The request methods are GET, POST, PUT and DELETE and has similar but not equal definitions as in the HTTP protocol. The response classes, success, client error and server error, have all their own set of response codes. As the methods they are also similar but not equal to response codes of the HTTP protocol.

The last part of the header holds a message ID. This identifier is used to map a received Acknowledgement or Reset message to an outstanding Confirmable message and to detect duplicated messages of any type. This is an important function to get a reliability mechanism of the communication when using UDP.

Figure 4 shows an example with a Confirmable message and its Message ID together with an Acknowledgement message in return and a second example with a Non-confirmable message.

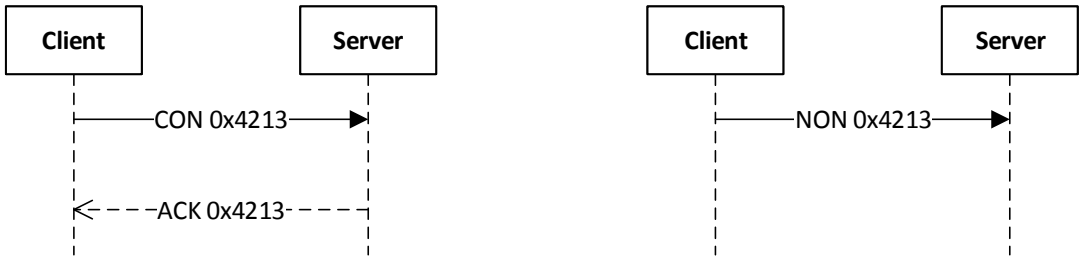

Figure 4: Example of a Confirmable message with Acknowledgement message and a Non-confirmable message.

2.2 Token value

Next to the header comes the Token value. The server shall echo the Token value in the response to allow the client to match a response to an outstanding request independently to the underlying message. The Token is therefore included by the server in either a piggybacked response or in a separate response that is sent later after the Acknowledgement message. The length of this field and the actual value is defined by the client.

In Figure 5 two examples are shown visualizing when the response is sent in a separate Confirmable message and when the response is piggybacked into the Acknowledgement message.

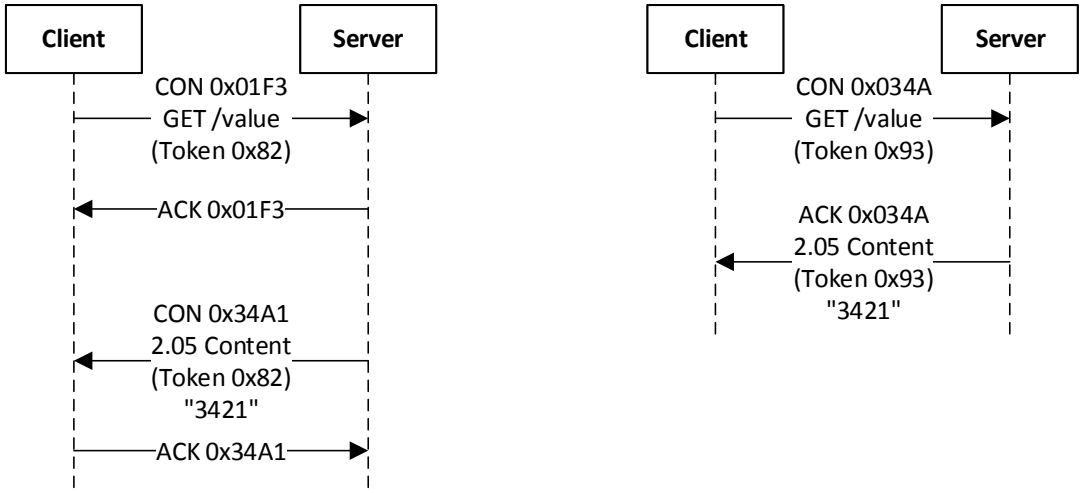

Figure 5: Example with separate response and piggybacked response

2.3 Options

A CoAP message contains also a set of zero or more options. Both request and response messages can contain options. An option included in a message defines the option number, the length of the value it holds and the value itself. The option numbers are defined by the CoAP specification. An option is allowed to occur multiple times in a single message. There are 15 options defined until now by the CoAP specification but there are five options of relevance for this paper (see Table 1). Those are used to specify the content-format of the payload data and to define the uniform resource identifier (URI).

Table 1: Options to indicate payload format and represent a uniform resource identifier of a CoAP request

Uri Option	Description
Content-Format	Indicates the representation format of the message payload
Uri-Host	Specifies the Internet host
Uri-Port	Specifies the transport-layer port
Uri-Path	Each instance specifies one segment of the absolute path to the resource to access
Uri-Query	Each instance specifies one argument parameterizing the resource

CoAP specifies seven different content-formats. The available formats are text/plain, UTF-8, link-format, XML, octet-stream, Efficient XML Interchange (EXI) and JSON. This paper utilizes only the octet-stream format as CIP is a byte-oriented protocol.

An URI is built up by a host, a port, a set of path segments and a set of query arguments. All options are represented by strings. By separating the URI into multiple options less parsing is required by the server. Figure 6 visualizes an example of a complete URI.

```
coap://device.domain.com:1234/my/path?abc=123
```

Figure 6: A uniform resource identifier including a host, port, path and a query.

2.4 Payload

If the last option is followed by a specified payload identifier (0xFFh) there is also a payload available. The payload takes up the rest of the datagram size. The format is specified by the Content-Format option.

3 New CIP network adaptation

To adapt CoAP to the CIP protocol it needs to handle explicit messaging and IO connections. An explicit message addresses a resource of CIP (object, instance, attribute) and applies a service to it. An IO connection continuously exchanges control and status data between the scanner and the adapter device. This could be done cyclically or on a change-of-state basis. In this chapter a proposal is presented of how CoAP can be used to handle transportation of CIP messages in terms of addressing resources, applying services, handling response codes and exchanging control and status data. The CIP protocol itself is not affected but it will be transported in CoAP messages. Figure 7 compares the protocol layers between EtherNet/IP and EtherNet/IP over CoAP.

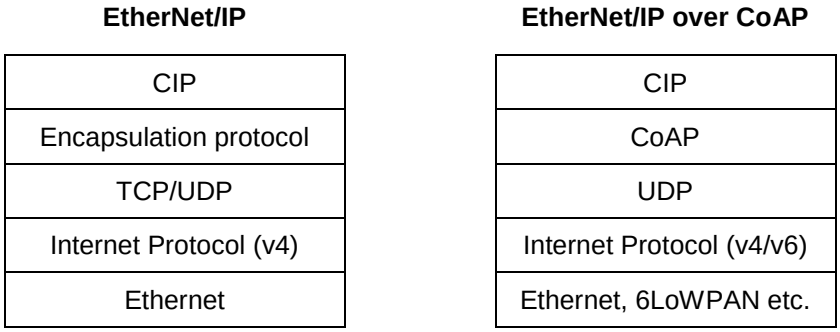

Figure 7: Protocol layering in EtherNet/IP and EtherNet/IP over CoAP. CoAP is designed to be used also over Internet Protocol version 6 (IPv6), this in beneficial for when the EtherNet/IP protocol is adapted to also support IPv6

As both EtherNet/IP and CoAP are based on standard TCP/IP communication, devices on both protocols utilize IP addressing and TCP/UDP data transports. The differences start when it comes to sending data to a CIP device and addressing its resources.

EtherNet/IP implements the Encapsulation protocol to handle messaging over TCP and UDP. This protocol offers a basic set of commands to retrieve identity and services of the device, to manage sessions and to send data. The lifetime of a session can span over multiple encapsulation commands.

When replacing the Encapsulation layer CoAP needs to deal with this functionality. CoAP has support for a discovery function allowing clients to find devices with multicast and get details about what services and resources it implements. Unfortunately, there is no support to manage sessions in CoAP, it only manages single message transactions. The Encapsulation protocol offers two commands, SendRRData and SendUnitData, to transfer unconnected and connected messages respectively. In CoAP there is only one type of messages and they are similar to the unconnected messages of EtherNet/IP. There is no support to handle connected messages in CoAP, but there are possible substitutes discussed in this paper.

The following subchapters will present and discuss solutions on how to use CoAP to communicate with resource constrained devices on a CIP network. However, it has been left out of the scope of this paper to investigate if the discovery function of CoAP is sufficient to replace the discovery and identity commands of the Encapsulation protocol.

3.1 Addressing resources

A resource in CIP is addressed by a class, an instance and an attribute number. This hierarchy of resources can be described well by using multiple instances of the path segment option of CoAP. By identifying the objects, instances and attributes by their assigned numbers short and efficient URIs can be created. In Figure 8 an example is shown where the URI addresses the TCP/IP Interface object (0xF5h), Instance 1, Host name attribute (6).

```
coap://cipdevice.domain.com/245/1/6
```

Figure 8: Uniform resource identifier addressing the TCP/IP Interface object (0xF5), instance 1, Host name attribute (6)

By parsing the URI segment options of the CoAP request, the device can internally generate the request path of a message router request.

3.2 Methods

CoAP offers four methods to interact with the resources of a device. The services offer clients to create and delete, retrieve and set resources. Table 2 describes the methods in more detail. From the CIP network point of view, which offers an extensive set of services, this could be a limitation of CoAP when integrating it to a supported CIP network.

Table 2: Description of CoAP methods

CoAP metod	**Description**
GET	Retrieves information from a resource
POST	Processing the enclosed representation in the request. The actual function performed is dependent on the origin server but usually a new resource is created or the target resource is updated. This method is not idempotent.
PUT	Updating or creating a resource identified with the request URI with the enclosed representation. This method is idempotent.
DELETE	Deletes the resource identified by the request URI

One could question if this limited number of methods are enough to control a CIP network device. Even if CoAP offers a very limited set of methods they are generic and apply well to the object model of CIP. The basic control of a CIP device is normally to retrieve and set attribute values of object instances but also to create and delete object instances. The CoAP methods could be translated to CIP services according to Table 3. The GET and PUT methods could translate to different CIP services depending on if the URI points out an attribute or an instance.

Table 3: Translation of CoAP methods to CIP services

CoAP method	**CIP service**
GET	Get_Attribute_All / Get_Attribute_Single
POST	Create
PUT	Set_Attribute_All / Set_Attribute_Single
DELETE	Delete

If there would still be a need to support other CIP services or object specific services, this could be solved by adding some kind of CIP service tunneling over CoAP or implementing new objects or attributes which would perform the same action but by using the methods of CoAP only. To allow usage of any CIP service in a CoAP request a new CoAP option could be defined. An advantage to use the CoAP methods only is that it would be sufficient to access and configure a device with a standard CoAP client.

3.3 Response codes

CIP offers an extensive set of status codes to describe success and different kind of erroneous behaviors. It also offers extended status codes to further describe the general status in an object specific point of view.

CoAP has a different approach with classes that separates the response codes between success responses, client error responses and server error responses. For every class a set of detailed response codes are defined.

Two approaches to handle CIP General Status Codes on CoAP have been investigated:

- Use the defined response codes of CoAP and define a complete translation table.
- Define CoAP options for CIP General Status code and CIP Extended status code.

Keeping to the features of CoAP as far as possible makes it easy to integrate with devices using standard CoAP clients. But when it comes to response codes the CoAP and CIP protocols have some differences. While CoAP differentiates its response codes in three classes, CIP has one single set of them. CIP offers also extended status codes defined per object where there is a need to do so. The paper has not looked into this issue in detail to see if there is a possibility to setup a translation table. However, there are situations where multiple response codes of one protocol translates to a single response code of the other protocol. An example of this is the success response codes of CoAP that only has one equivalent on CIP. Translating multiple response codes into a single one would be possible but translating in the other direction is not as simple. There is also no good way to integrate the extended status codes of CIP when using only the message code field in the CoAP header.

The other alternative is to define CIP specific options on CoAP which hold the General Status code and the Extended Status code of CIP. This would allow the CIP status codes to be transported transparently over CoAP. The bottleneck is that a standard CoAP client would not be able to understand the option content, it would only be able to parse the option format itself. But the CoAP message code field could still be used to signal a success or an error response by setting the message code class (success, client error or server error). This would make it possible for a standard CoAP client to at least notify about error responses. For further details the user has to interpret the options holding the CIP status codes.

3.4 IO connections

CoAP is by design a messaging protocol with requests and responses. It does not include cyclic data exchange similar to a class 1 connection on EtherNet/IP. An alternative to cyclic data exchange for resource constrained devices is to let clients register for data updates.

CoAP offers two different techniques to accomplish this:

- Observe resources
- Publish/subscribe to topics linked to resources

The “observe” resources feature is specified as an extension to CoAP [4]. It utilizes normal CoAP Get requests with responses, but with the addition of a CoAP option named Observe Option. This option is specified to hold an unsigned integer with the length of 0-3 bytes. In a request the Observe Option extends the request to not only retrieve the current value but also indicate that the client is interested in future updates. The option shall be assigned the value 0 or 1 to register or deregister the observation of a resource. In a response the Observe Option indicates that it is a notification of an updated resource and it is assigned a sequence number for reordering detection.

When a client wants to observe a resource of a CoAP server device it sends a normal Get request that points out the resource but with the addition of the Observe Option. The server will respond normally with the current value of the resource but also save the client as an observer. Whenever the resource gets updated, notifications with the update value will be sent to the observing client. The updates are sent as subsequent responses to the initial request with the Observe Option present to identify the response as a notification.

The other approach is about a publish/subscribe mechanism that is under development and is also supposed to be an extension to CoAP [5]. This technique would make it possible for the scanner to subscribe to data updates and receive data from a device every time it changes. In the same manner a device could subscribe to data updates from the scanner to retrieve control data.

The publish/subscribe mechanism of CoAP requires a broker device to manage publications and subscriptions. To be able to publish data a device first needs to create a topic on the broker. A topic is a unique string conventionally formed as a hierarchy, e.g. “/device/path/to/value”. Other devices subscribe to this topic to receive future data updates. Devices can either rely on an external device serving as a broker (Figure 9) or implement a broker themselves to get a “brokerless” configuration (Figure 10). If an external broker is used the device which shall publish data creates topics in the broker device while the scanner or some other client subscribes to those topics. In a “brokerless” configuration the device itself offers a pre-configured set of topics which other devices can subscribe to.

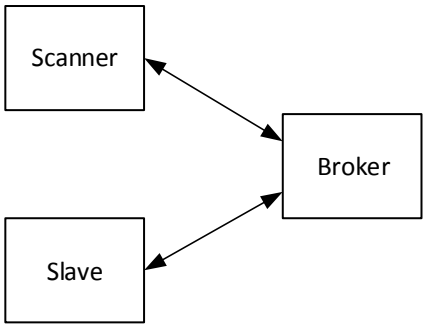

Figure 9: Setup with external broker

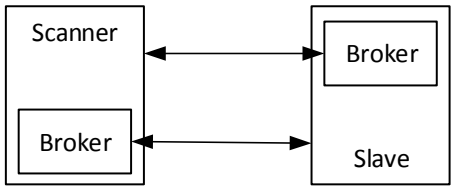

Figure 10: "Brokerless" setup

Topics on CoAP are constructed by URI-path segments. Multiple segments create a path pointing out a resource. On CIP a topic could point out for example an assembly instance by addressing the assembly object and one of its instances.

3.5 Security

The standard implementation of EtherNet/IP today has an extension for enabling secure communication, known as CIP Security. The historical view of security of an industrial control network has been via a secure in depth perspective where the control protocol itself is not secure but the environment around it makes it secure. For instance, a control network may not be connected to the enterprise network at all or via firewalls and routers to ensure that no one from the outside can access devices in the OT network.

With the development of Industrial IoT the need to open the barriers between the IT and OT networks is needed to be able to exchange data between these two worlds. Introducing the Ethernet protocol to small sensors and actuators that historically have been connected via discrete/analog in or outputs or via other protocols like IO-Link also affects the secure in depth concept, making it easier for an outside person to fetch information from a sensor or even control an actuator remotely.

CoAP has built in support for communicating over Datagram Transport Layer Security (DTLS) which is also used for securing the UDP communication in standard CIP Security over EtherNet/IP. DTLS offers a standard and secure way to handle key management, data encryption, message integrity and authentication. For authenticity the CoAP protocol has specified three options; Pre-shared key, asymmetric key-exchange using X.509 certificates or asymmetric key-exchange using raw public key. For the asymmetric calculations the support for Elliptic-Curve DH (secp256r1) is required, this cipher is also

required by CIP Security. After a successful key-exchange the DTLS protocol uses symmetric cryptography to ensure message integrity, CoAP requires support for AES128 cipher which is also required by CIP Security. As CoAP is using the same techniques for securing the communication as CIP Security it will be relatively easy to integrate DTLS support for CoAP into the current specification.

With a goal to be able to produce cheaper and smaller applications it might be contradictive to add cryptographic functions that will require more computer power and more memory from the host system. The commercial IoT solutions targeting our homes are however facing the same problem, this has led to simple Microcontroller Units with embedded hardware accelerated cryptographic functions available on the market at reasonable prices. Using systems like these will not require the CPU to be much more powerful to implement the security features, it will however add to the total cost of the system.

CoAP is also suggesting the use of raw public keys over X.509 certificates, as defined in RFC7250 [6], to reduce complexity in parsing and validating data of these certificates. In the raw public key approach a device is given a unique asymmetric key pair during production, the public key is used to calculate a hash identifier as defined in RFC6920 [7]. During commissioning of a network this identifier is loaded into the other end point of the connection by for example scanning a bar code on the side of the product or by manually entering it in a human readable format.

4 Example setups

There are numerous ways to connect an EtherNet/IP over CoAP adapter to an EtherNet/IP network. This chapter shows some examples on how an EtherNet/IP over CoAP device can be connected to an industrial network using different EtherNet/IP scanner implementations and physical layers.

Figure 11 shows a regular EtherNet/IP network with a standard EtherNet/IP scanner. To be able to connect an EtherNet/IP over CoAP adapter a gateway device that translates regular EtherNet/IP requests to EtherNet/IP over CoAP and vice versa is needed. This type of gateway enables the possibility to retrofit an EtherNet/IP over CoAP adapter into a regular EtherNet/IP network. A standard CoAP client can from either inside the OT network or the IT network via a router fetch data directly from the EtherNet/IP over CoAP adapter.

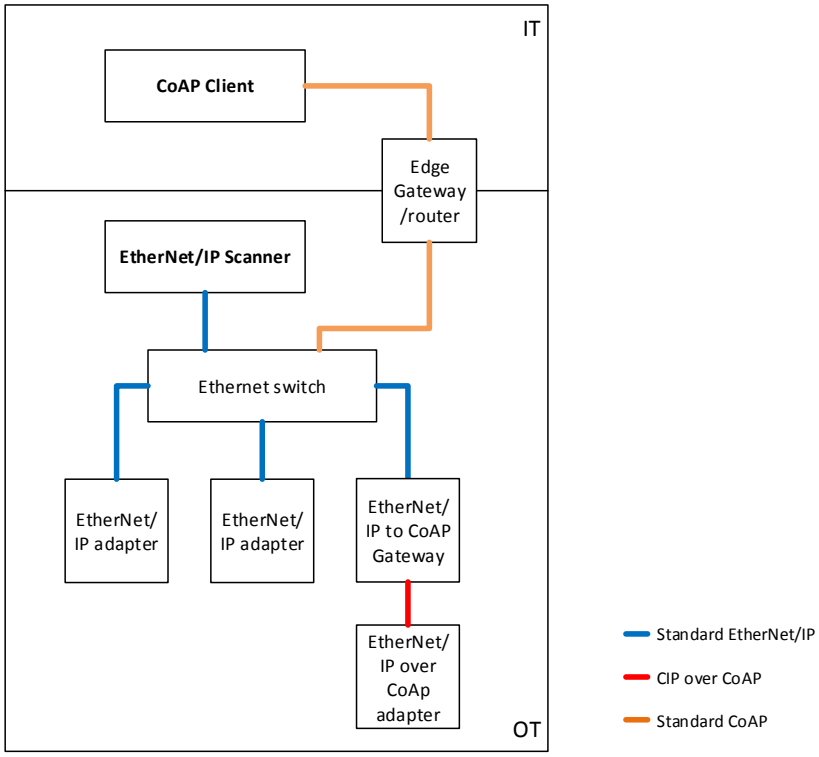

Figure 11. Example network with an EtherNet/IP to CoAP gateway device that translates standard EtherNet/IP to EtherNet/IP over CoAP. This type of gateway enables retrofitting of EtherNet/IP over CoAP adapters into a regular EtherNet/IP network. A standard CoAP client implementation can be used from either inside the OT network or from the IT network via a Gateway or Router.

Figure 12 illustrates a network with an EtherNet/IP scanner that has been enabled with support for EtherNet/IP over CoAP. With this type of scanner EtherNet/IP over CoAP adapters can be connected directly into the standard Ethernet network. The Ethernet switch in this example also has support for single pair Ethernet, this gives the possibility to connect an EtherNet/IP over CoAP adapter communicating over single pair Ethernet to the standard Ethernet network. By adding a Wireless access point to the network it is possible to connect multiple wireless sensors to the network.

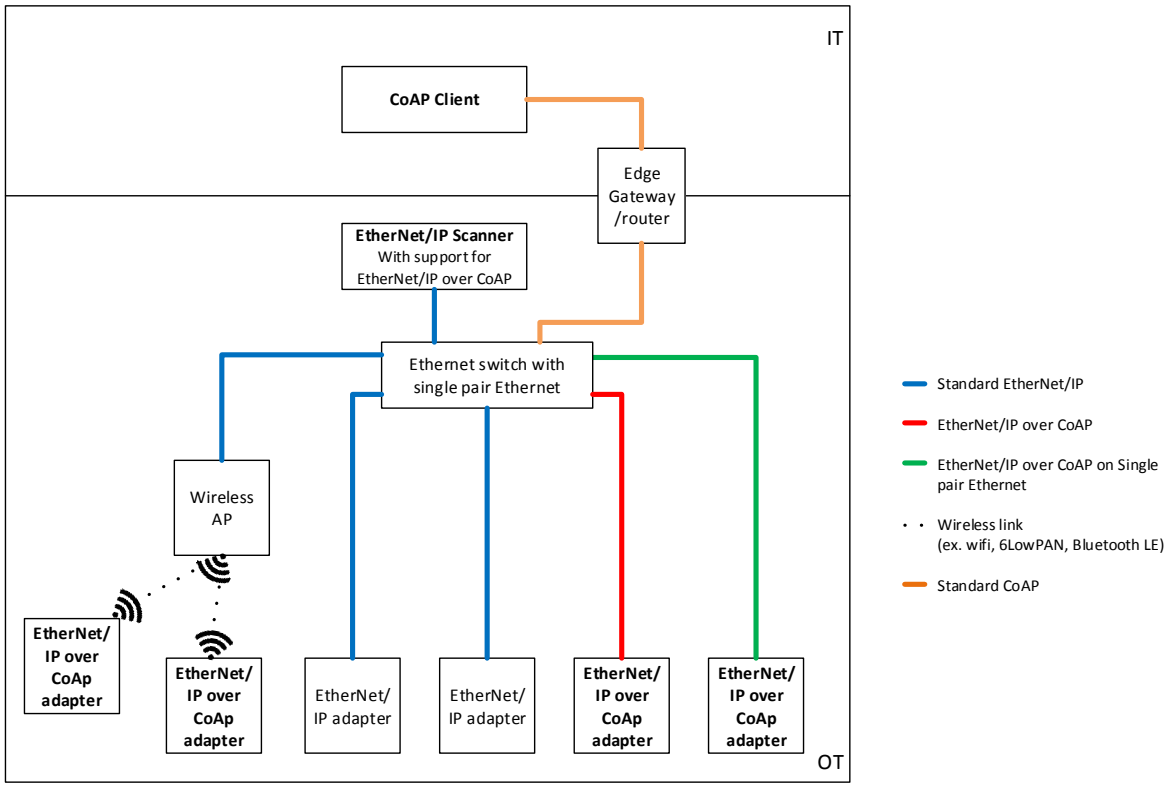

Figure 12. Example network with an EtherNet/IP scanner with built in CoAP support and EtherNet/IP over CoAP adapter connected directly to the Ethernet network via standard Ethernet, single pair Ethernet or via a wireless connection.

5 Conclusion

The Internet of Things has really enhanced the development of new technologies targeting resource constrained devices the past years. New communication technologies with reduced overhead and complexity open up possibilities for devices like simple sensors and actuators with constrained resources to be connected. The Constrained Application Protocol (CoAP) is such a protocol which has succeeded to reduce the communication overhead and achieve low complexity without compromising the reliability of the network for certain use cases. This paper has investigated CoAP and how it could be adapted to CIP to connect resource constrained devices. Compared to previous work of this area within ODVA, that has focused to achieve the needs of resource constrained devices by optimizing the EtherNet/IP protocol, this paper has chosen another approach by taking advantage of an already developed IoT protocol. The paper has focused on how to address CIP resources, apply actions to those and how to exchange IO data by using CoAP as the underlying transport protocol. Also security has been considered.

The investigation shows it would be possible to use CoAP to connect resource constrained devices to a CIP network even if doing so sometimes requires new approaches to achieve the functionality of a CIP network. CoAP as a messaging protocol conforms very well with the explicit messaging of CIP. To address resources of the CIP object model the address scheme of CoAP works without any limitation. With the help of additional CoAP Options it could also be possible to forward standard CIP services and status codes in the CoAP requests and responses. However, CoAP does not conform very well to the IO data exchange of CIP. To solve this, the paper proposes to make use of either the observe resource or the publish/subscribe feature of CoAP to replace the cyclic data exchange. With no direct support for cyclic IO data exchange CoAP shows it is not designed with the industrial communication model in mind, but designed to achieve the needs of resource constrained devices. CoAP does not fit perfectly in to the communication model of CIP out-of-box, but to allow simple devices to connect to CIP networks it may also be required to adapt CIP to the needs of these devices.

There are still many aspects to look into before saying it is possible to use CoAP as a CIP network adaptation. The techniques proposed to accomplish IO data exchange over CoAP and how to map these to the object model of CIP, with the Connection Manager and the Connection Object, must be examined. Also the possibility to add additional CoAP Options for CIP services and CIP status codes must be investigated. With these areas open for further investigations, and if other vendors of the ODVA community see potential in this concept, it will hopefully be possible to coordinate and together continue the work of this paper.

Regardless the outcome of the adaptation of CoAP to EtherNet/IP, the approach of this paper to integrate resource constrained devices to CIP hopefully contributes to the future solution of connected resource constrained devices in CIP networks.

6 References

[1] D. Xu, P. Brooks. “Extending EtherNet/IP™ to Resource-Constrained Industrial Things”. 2015

[2] H. Wernersson, Y. Atwa. “Resource constrained Industrial IoT device”. 2016

[3] Z.Shelby, K. Hartke, C.Bormann. “The Constrained Application Protocol (CoAP)”. RFC 7252. June 2014.

[4] K. Hartke. “Observing Resources in the Constrained Application Protocol (CoAP). RFC 7641. September 2015.

[5] M. Koster, A. Keranen, J.Jimenez. “Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)”. draft-koster-core-coap-pubsub-05. July 7, 2016.

[6] P. Wouters, H. Tschofenig, J. Gilmore, S. Weiler, T. Kivinen. “Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)”. RFC 7250. June 2014.

[7] S. Farrell, D. Kutscher, C. Dannewitz, B. Ohlman, A. Keranen, P. Hallam-Baker. “Naming Things with Hashes”. RFC 6920. April 2003.

****