Design for Internet of Things
Prof. T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science-Bengaluru

Lecture - 36
COAP – 01

Let us learn one more protocol for IoT systems. When you have lot of data that is collected by systems, and you want to sort of use this data, you want one machine to read data, and you want the other machine to consume the data, right? So we have been saying this machine to machine is very important.

Humans not being the loop is also very important. But if you look at the way humans have used computers, you know the famous browsers that we have been using, right? You have used Mozilla Firefox or you would have used Internet Explorer. In the previous 25 years back when we were kids we were using Netscape. Netscape browser was a very popular one.

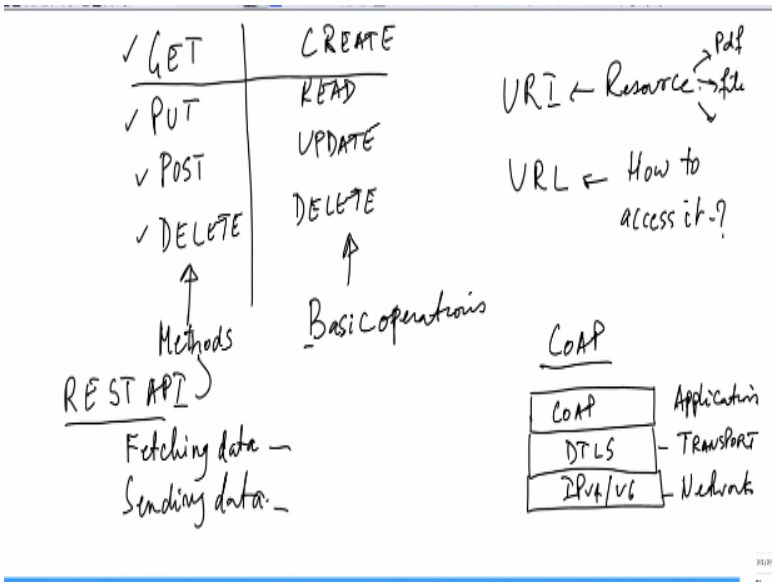
So there are different type of browsers, and Chrome is a browser, right? Google Chrome and so on. So lot of browsers come. And what do you do when you are in front of a browser, you would type in a URL, you type in something you would say Google or you will say HTTP, HTTPS, colon double slash in the address bar, you would type it and then you would go access some resource and come back, okay.

All this you have been doing without realizing what is actually happening behind, okay. Let me point you to something that will give you in a concise way what exactly is the paradigm of fetching data, requesting for data and fetching that data. That actually comes under what is known as a paradigm called REST architecture, representational state transfer, REST architecture.

This was proposed by a PhD student during his years of working by name Roy Fielding. You please Google it, you will find his name. It is I think, chapter five of his thesis, you can actually download and understand REST architecture, okay. Like publish subscribe, there must be some architecture even here, right? And this architecture is indeed the REST architecture.

Now this REST architecture, supports what are known as methods, okay. And these methods essentially lead to certain operations. Let me point you to what I wrote here.

(Refer Slide Time: 03:02)



You want to fetch data, you want to send data. For that, you use this REST API, which supports these methods. What are the methods? GET, PUT, POST, DELETE. These are the methods that you apply. And when you apply these methods, you may be able to operate on getting the data or adding the data, appending the data, creating new data, deleting the data, and so on, and so forth.

For example, if your method is DELETE, what will happen to the data that is there? It will get simply deleted. If you PUT, you are going to essentially create data, you are going to add something to it. If you POST, you may be able to append something to it. If you say PUT, it might replace. Whatever is there may be gone and it may get replaced with something else. So there are differences, nuances in its implementation.

You can read them in detail and get a super understanding of all of that. Now let me come back and connect to another important point. When you type in HTTPS, secure HTTPS colon double slash, let us say www.iisc.ac.in. That is where I come from, right? So if you type in that, what actually you are doing is you are just typing that. What actually happens is you are invoking a method called GET.

When GET method is invoked, the server gets it. The IISc server says, oh, this guy is looking for that material, and then transfers that material for in response to GET, okay. So in other words, that is what is actually happening. You will also be able to get a form sometimes. And you will fill up fields in that form and you will press submit.

Submit means what? You must be doing one of the other operations, other methods. That method could either be POST or PUT. Please find out the difference between POST and PUT. I am not going to elaborate. But it is up to you, you have to know these little nuances, okay. So all that is actually happening with those REST API methods, okay. In other words, think about a column, which has, in fact two columns in a table.

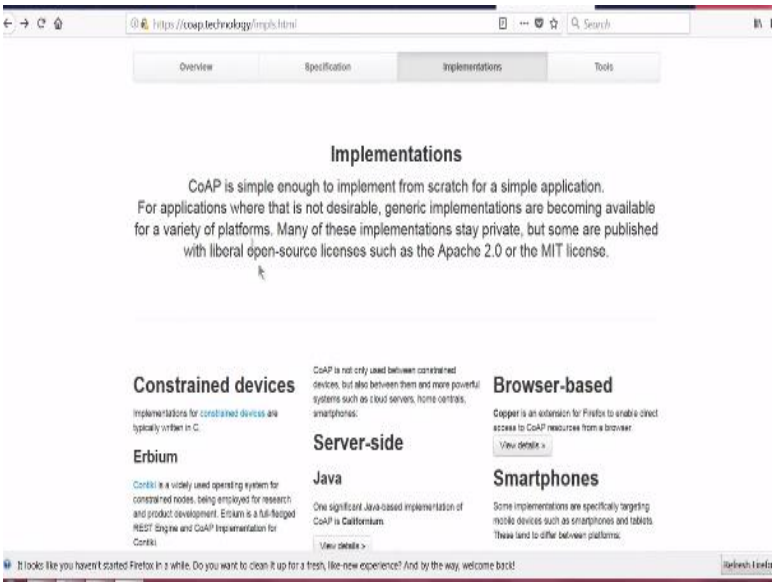
One column comprising of the methods and the other column comprising of operations. The operations are CRUD; CREATE, READ, UPDATE, and DELETE, second column. First column is GET, PUT, POST, DELETE, okay. And you can have other things also, but I have broadly captured. What maps to what? First column, what does it map to the second column? Figure out. You will get to understand REST architecture very well.

But you may now ask one important question, which can be very strikingly difficult to appreciate. Sir, you are talking of a computer, a laptop, or a mobile phone, or a handheld, where you type in http blah, blah, and then you get some data from the server. These are devices which are having battery, big batteries, having large systems, which can live for longer times, maybe they have a charger port, where you can connect, and then you can do a few things.

How does it work for IoT? Yes, it does not, that will not work for IoT. Because they are, IoT we studied so much about battery, battery life, and surviving on small coin cells, energy harvesting, and so on. So it is not going to work. But nothing stops you from taking the principle of REST API and adapting and coming out with another protocol, which is best suited for IoT devices. Yes, you got it right.

Then came the birth of this new protocol called CoAP, constrained application protocol. Exactly like REST API, but not heavy like HTTP. Not heavy like the ones that you know or I know of. Meant for constrained devices, constrained application protocol. So you may ask, are there implementations? Yes, there are plenty of implementations and I will show you a snapshot of the implementations.

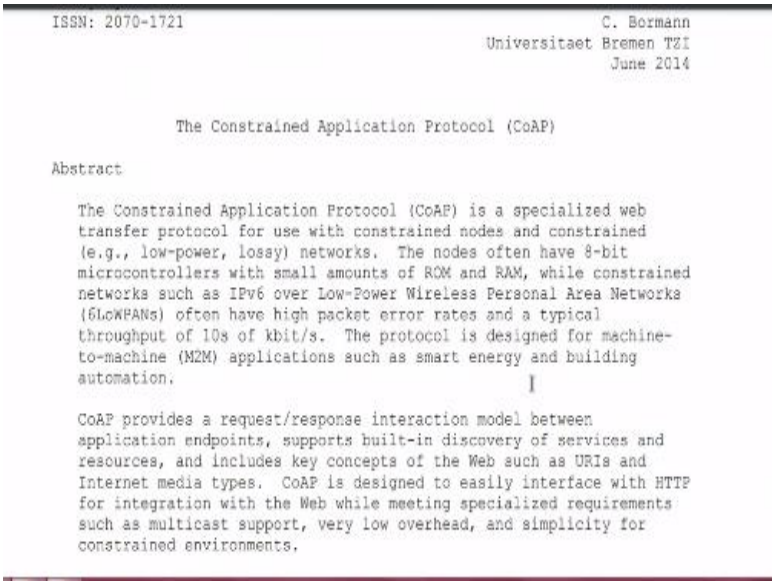
(Refer Slide Time: 08:05)



These are the implementations of CoAP. CoAP is a simple enough to implement from scratch, for a simple application. For applications where it is not desirable, that is not desirable, generic implementations are becoming available. You already have existing implementations. If you do not want to implement your own, use the existing one. If you want to develop your own, very simple, what should you do?

How do I start? Nothing, just go and read this RFC.

(Refer Slide Time: 08:35)



ISSN: 2070-1721

C. Bormann
Universitaet Bremen TZI
June 2014

The Constrained Application Protocol (CoAP)

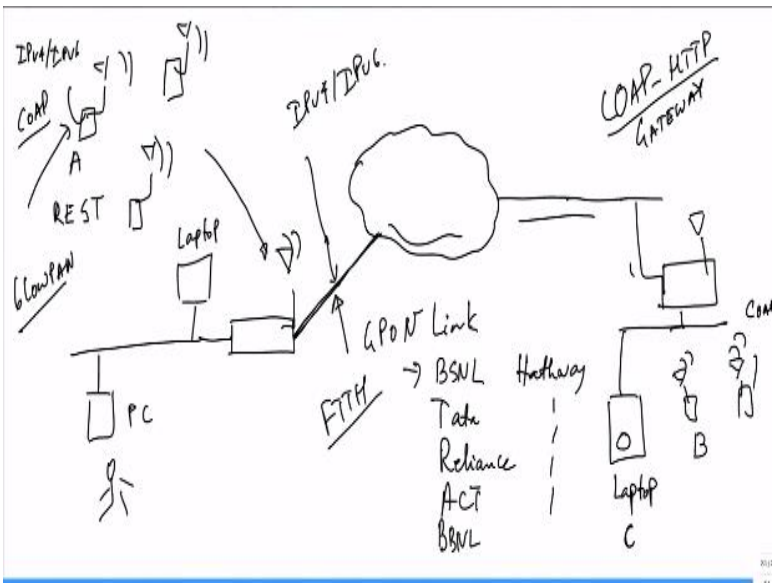
Abstract

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) often have high packet error rates and a typical throughput of 10s of kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.

CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments.

Essentially pick this RFC and read it. That is all. This RFC is indeed the source for anything. You do not have to implement everything. You may want to implement only the bare bone that may be required for working. But see the beauty. You are using REST API and you are still using it for constrained environment. What is the specialty of this is something that may be bothering you. For that what I will do, I will show you how you can actually interwork.

(Refer Slide Time: 09:11)



Think about the internet. Think about your network, okay. This is a PC. This is the gateway. This is a gateway, which also has a wireless router for several small devices in your home. These are all wireless devices and this is a wired interface. You have PC, you perhaps have laptop, and TV and so on and so forth, which are all connected to the, this is your internet link.

It could be a PON right, GPON. You have people like BSNL, then you have TATA right, then you have Reliance, then you have ACT, you have BBNL, you have Hathway. I am just putting down so many companies which offer you fiber to the home, right? Fiber to the home FTTH, fiber to the home link, okay. Now somewhere here is another network and there, there is maybe another gateway device.

And then there is a wireless router and there are small nodes here, okay. Just shown a small picture of okay, and show the antenna, and then I must show them radiating. So let me put the antenna and then they are radiating, okay. Let me just take only two for a simple thing. Let us say this A wants to send data to B. And look what is happening. These are two sensor nodes, okay.

These are two sensor nodes. And they are not really the mainstream kind of nodes like laptops and PCs and so on. But it works, this simply works, okay. How? Because this is also talking RESTful. These are all talking REST architecture, all of them use the REST API. And data simply goes, comes wirelessly here. You may convert this into a IPv6 or IPv4 packet, put it out so it is going out here.

Or as IPv4 or IPv6 packet. In fact, IP part is interesting. It can actually be implemented here also, okay. Here you can have the IPv4 stack, or IPv6 stack. Also on the node. There is nothing stopping you from doing it. There are implementations of IP stack for constrained environments, okay. So you can use those stacks also. I will not get into the detail. But anyway, it goes as an IP packet. Goes as a CoAP packet.

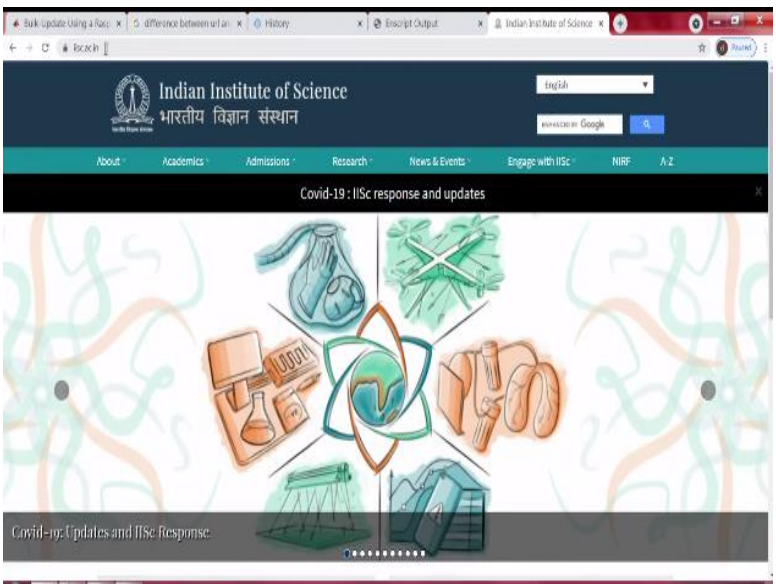
Travels all the way through the internet, comes here. And since this can talk CoAP and this also is talking CoAP, you can see that you can get data from source to the destination across the internet. So nothing stops you from using these systems widely spread out clearly using CoAP protocol. So if you have to use CoAP, what does it mean? It means that you are using CoAP in the application layer. You are using datagram transport layer security, which means packets can be encrypted using DTLS.

And you can use IPv4 IPv6 transport, and push this packets on wireless, wired links up to the nearest gateway, and go on the core internet and then go away from there. So the reachability of these nodes is exhaustive. But folks, let me ask you another question okay, which may be of interest to you. Supposing this CoAP A wants to talk to C, okay. This is the network. And there is a node which is a laptop, okay.

A wants to talk to laptop C. Would this work is the question. Yes, it will work because you have to build on this router CoAP to HTTP perhaps. That is all. One gateway device you have to build. Very simple to map CoAP to HTTP because it also uses the REST architecture. So it becomes very simple. And then whatever resource is here can also be transmitted back to this A.

How, because the way resources are identified, and the way the resources are addressed, uses the same REST architecture. To give you an example, you know about the famous URI and URL. URI is about a resource like a PDF or a file or whatever. And how to access this URI, URL will be used. And where do you type the URL? You are typing the URL in the famous browser window here, folks. This is the URL.

(Refer Slide Time: 15:42)



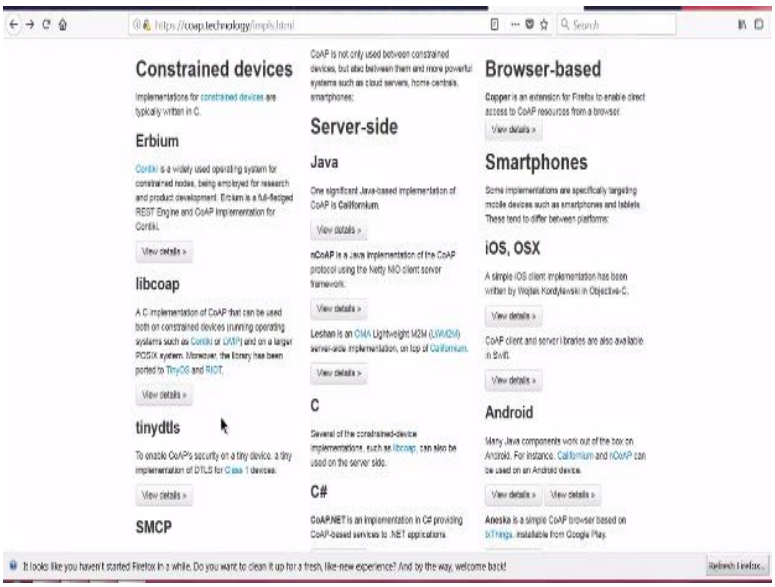
This is what you type here. Whatever you type here, whatever you type here is the URL, okay. There you are. This is how it is going. You have typed the URL, you have asked what you are looking for, and the address, then it will show you the IISc page.

Supposing you are interested in accessing a particular part of the page. That means only that resource, then you specify the link. It is coming to the main page now, okay.

If you qualify it more, in this itself, you quickly realize that you can actually access that specific resource. So what does it mean? Yes, you can access any URI, of course, I assume that permissions are there. And you can specify that in the URL. And then you can access that resource. CoAP uses the same thing that HTTP uses. Because HTTP is also based on REST architecture, it uses the REST API.

So folks, this is really the basic of CoAP protocol. You can use several implementations. Let me show you a snapshot of that.

(Refer Slide Time: 17:13)



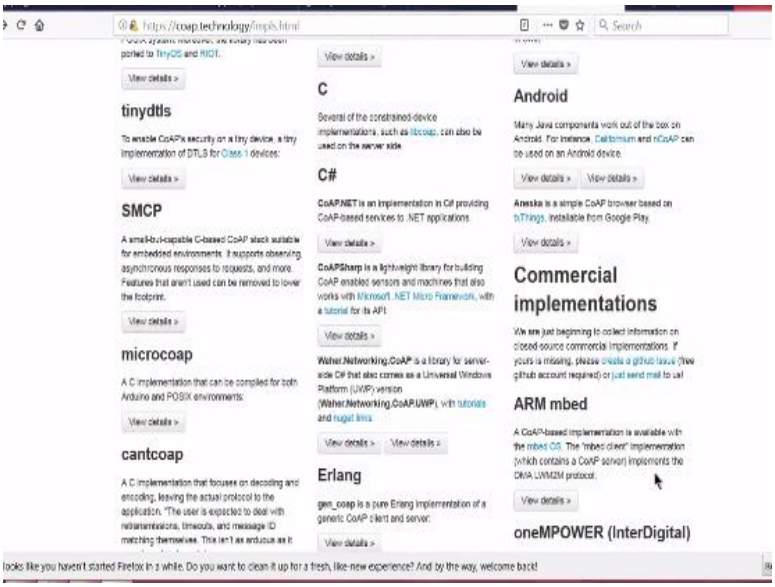
Constrained devices, CoAP on constrained devices, Erbium, libcoap, tinydtls SMCP, microcoap, cantcoap, Lobaro CoAP. Lobaro CoAP is good, because the express IF's, ESP8266 is a very powerful module, okay. And it is an MIT licensed C implementation for a constraint, for certain set of constraint devices, which use these protocols like CoAP. Particularly observe is a very powerful feature of CoAP.

We will come to that. And there are lot of demos. So I would go and vouch for you to check out on a Lobaro CoAP, Because it is there, out there. And it is a very powerful popular module, okay ESP8266 is a very powerful module. You can buy it for 200 rupees folks, as cheap as 150 to 200 rupees, right? So buy it and try it that is important, okay.

Server side, Californium. Then Leshan, C, C#, Erlang, Go okay, Python. So many implementation. Ruby, you can have Ruby implementation of CoAP also. Rust implementation, and so on. It is also available for browser based access of resources. This is possible using Copper. You have a Mozilla Firefox and you go and pick up Copper, you will be able to access, direct access to CoAP resources from within the browser. Why?

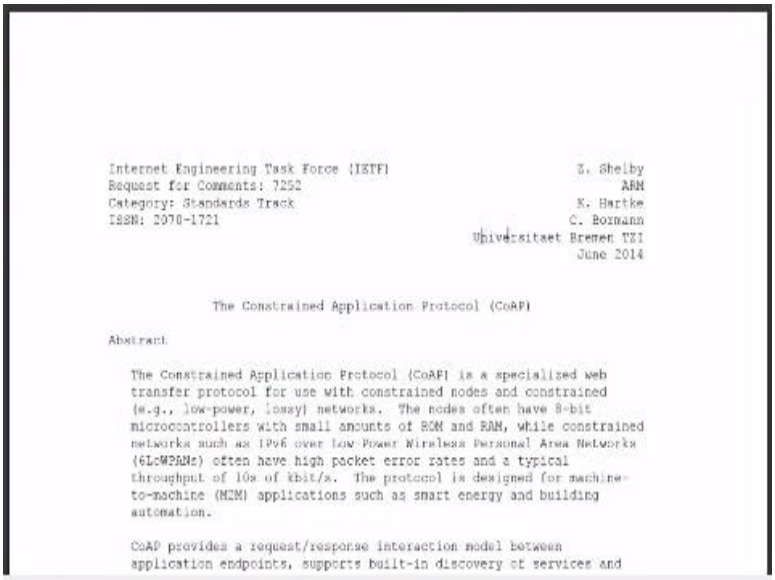
Because browser also supports RESTful architecture. CoAP also is RESTful architecture. So access to resources of CoAP resources via the browser is straightforward. That is why you can use browsers also easily. Smartphones, iOS and so on and so forth.

(Refer Slide Time: 19:18)



ARM mbed also has an implementation, please look it up. Therefore, these are very popular and very useful. All right. Now we need to know a little more about the protocol before we move on and get some grasp on the protocol.

(Refer Slide Time: 19:39)



Internet Engineering Task Force (IETF)
Request for Comments: 7252
Category: Standards Track
ISSN: 2070-1721

Z. Shelby
ARM
K. Hartke
C. Bormann
Universitaet Bremen TZI
June 2014

The Constrained Application Protocol (CoAP)

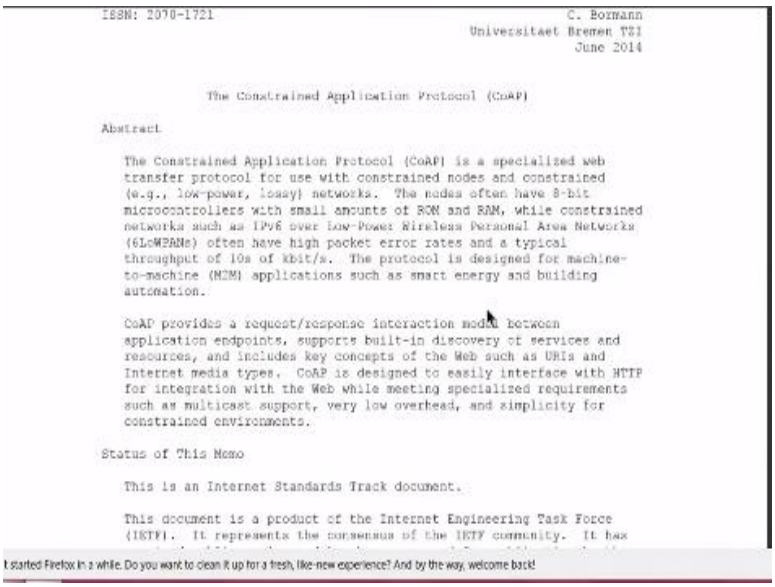
Abstract

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) often have high packet error rates and a typical throughput of 10s of kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.

CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and

So let us get into the details of this protocol. And I will explain the most important parts of this protocol. I am sure you will enjoy knowing lot about this protocol as we go along now. Let me start by telling you that you must read this 7252. This is in the Standards Track and it is offered by the IETF. And there are lot of videos by Zack Shelby from ARM. Do look up YouTube videos from Zack Shelby. It is explained pretty well.

(Refer Slide Time: 20:15)



ISSN: 2070-1721

C. Bormann
Universitaet Bremen TZI
June 2014

The Constrained Application Protocol (CoAP)

Abstract

The Constrained Application Protocol (CoAP) is a specialized web transfer protocol for use with constrained nodes and constrained (e.g., low-power, lossy) networks. The nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while constrained networks such as IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) often have high packet error rates and a typical throughput of 10s of kbit/s. The protocol is designed for machine-to-machine (M2M) applications such as smart energy and building automation.

CoAP provides a request/response interaction model between application endpoints, supports built-in discovery of services and resources, and includes key concepts of the Web such as URIs and Internet media types. CoAP is designed to easily interface with HTTP for integration with the Web while meeting specialized requirements such as multicast support, very low overhead, and simplicity for constrained environments.

Status of This Memo

This is an Internet Standards Track document.

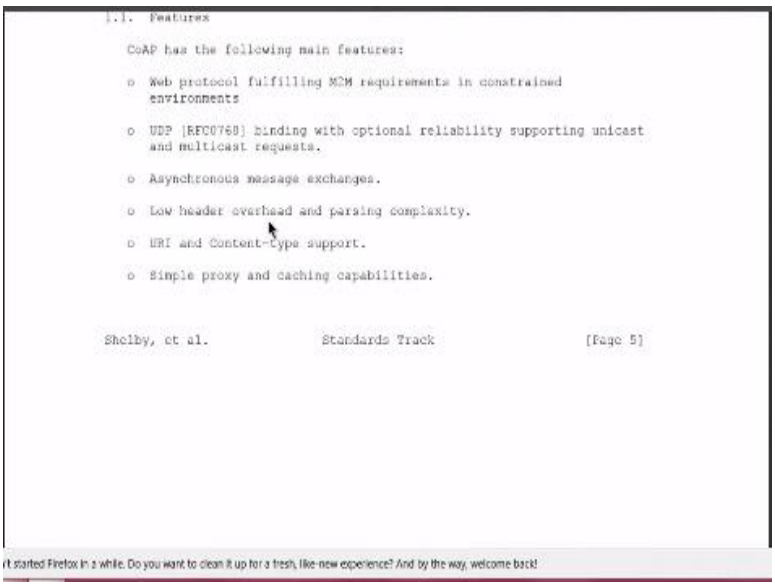
This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has

t started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

Now what is CoAp? You will now get the definition of this. It is a web transfer protocol. It is used for constrained nodes, low-power and lossy networks. 8-bit controllers also will support. 16-bit will support. IPv6 implementation on embedded platforms are possible. Then you have something called 6LoWPAN. I think I mentioned to you about IPv6 implementation on these nodes.

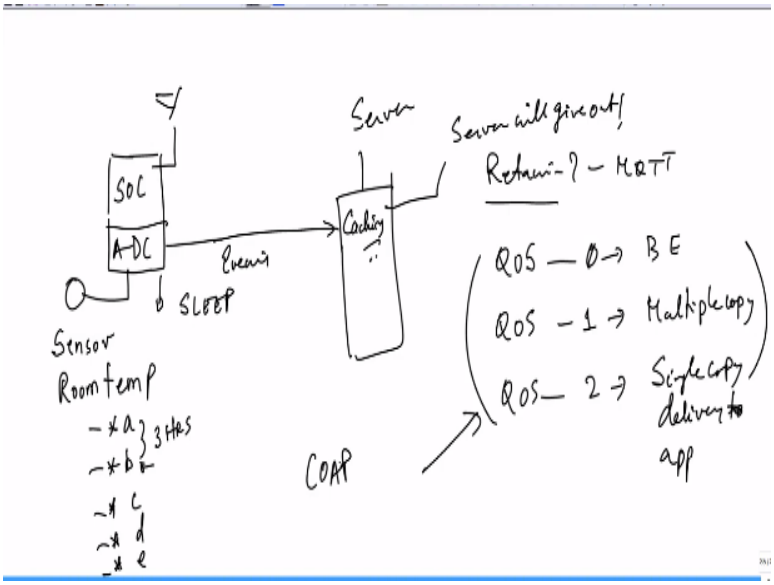
This can be a 6LoWPAN implementation. This is nothing but IPv6 implementation for constrained environments. Full IPv6 stack is possible, okay. Then it tells you about that. And then it is telling you about several nice things about this protocol. We will go into some important aspects of this protocol. I would not read every line because it is going to take a humongous amount of time. But I will cover this protocol from the features, okay. Very important. So let us move on.

(Refer Slide Time: 21:31)



Okay, it uses UDP. It has low header overhead. URI and context type support and simple proxy and caching capability. See, HTTP to CoAP is very simple. You can do as simple conversion. So gateway functionalities between CoAP and HTTP are easily doable. Quite like that. If there is a resource that is cached, access to that cached results is also a possibility. Now this has a super bearing on super bearing on lifetime of sensor nodes.

(Refer Slide Time: 22:16)



Think about a CoAP node okay, which has connected to an ADC. It has an ADC. Let me draw it big so that you will be able to see it clearly on the screen. Okay, this is the SOC. To this ADC you have connected a sensor. Let us say this sensor is your room temperature, your room temperature. Folks, if you look at your room temperature in the morning, it will have some value ‘a’.

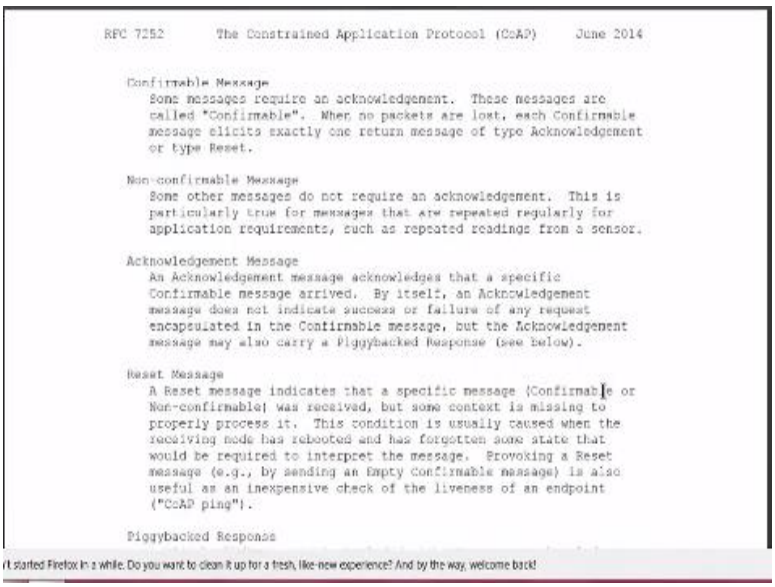
Okay, let me put it as small a. As it comes to noon, it will have another value ‘b’. Between morning and noon there will be about three hours I suppose. And similarly afternoon and similarly evening and maybe in the night, right? Only five times the value may change drastically. You will see five discrete change in values, okay.

If it puts out a value in the morning, if it puts out a value in the morning, you may want to cache it on a larger server, okay. And now if there is any request for this node for temperature, who will serve out? The server will send serve out. Is it not similar to Retain? Yes, indeed. Is it not something similar to Retain of MQTT? Yes, indeed. What is the advantage? You sleep now. You give out your morning value and go to sleep.

Come back when it comes to you know that it is crossed a certain threshold. Again, update noon, okay. Again update evening. And server, the cache will simply serve it out. So caching will work very well. So folks that is what this RFC is telling you. Simple proxy and caching capabilities. So every line here you have to think deeply folks, from your energy perspective.

It is not that MQTT and CoAP do not have security features. It is just that I did not get into the details. But they are very secure protocols, read them very carefully. And I am not spending time there. So he has definitions of endpoint, sender, recipient, client, server, and so on. Origin server, intermediary, proxy, forward proxy, reverse proxy, CoAP to CoAP, cross proxy, and so on.

(Refer Slide Time: 25:47)



Now we will come to messages. Folks, again you may want to ask the question, how is it going to be different from MQTT, which is obvious, correct. Therefore, you have this many messages, okay. We will get into each one of them in sufficient detail right away. Now if you take QoS 0 of MQTT. What does QoS 0 of MQTT take? So now you see already I mapped one of them to, here I mapped it to MQTT.

Now I am mapping one more to QoS 0. Recall QoS 0. This is best effort. QoS 1, multiple copies. QoS 2, single copy, delivery. I will say single copy, delivery to app. Quite similar you have, this is in the MQTT world. I just put back your memory. You have to do something similar in the CoAP world. And what are those? Very simple. Confirmable message. That means you are looking for a acknowledgement.

You may send a message, you may not want an acknowledgment. In which case you do non-confirmable. You do not require an acknowledgement. If you want an acknowledgement, do it with confirmable messages. If a packet is lost in confirmable,

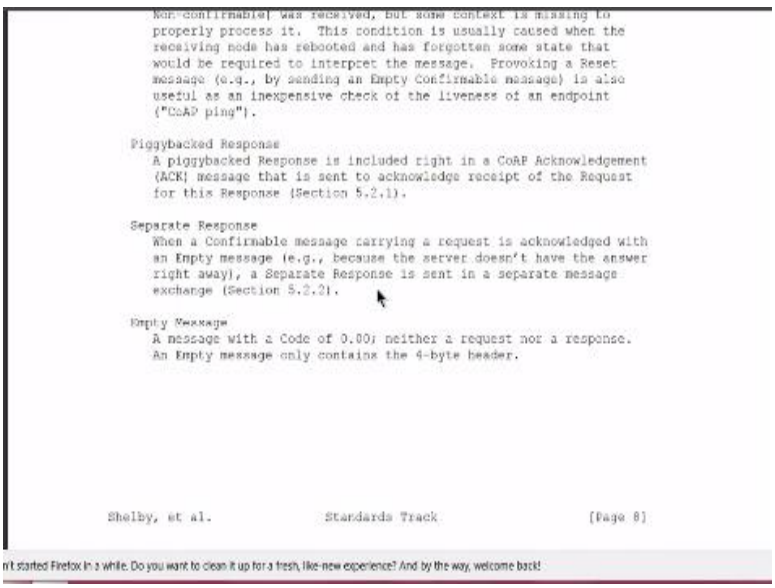
you will have to retransmit it. But when no packets are lost, essentially, there will be one return message which will give you acknowledgment, and says I got your packet.

And that is what is the acknowledgement messages. It is meant for confirmable messages. Reset message. Reset message says, hey, I got your packet. But I think there is something missing in what you asked for. Maybe you asked for a resource or a context, which is missing. I could not process it properly. Maybe you did not specify the URI properly. But I got your packet. This is important. I got your packet.

It is not that I did not get, I got it. But I could not serve you. Therefore I am sending you a Reset message. Now that is one possibility. Second possibility is that the receiving node for some reason ran out of battery. And people ran and replaced those batteries. It was perhaps one minute or two minutes delay. By that time it rebooted right, because it lost battery. It was rebooted, okay.

And therefore it says, you asked me something. I do not know what you asked me for. I have no knowledge of what you asked me. I am sorry. So it says this condition is usually caused when the receiving node has rebooted and has forgotten some state that would be required to interpret the message and say sorry, I do not get you. So I am going to send you a Reset message it says.

(Refer Slide Time: 29:59)



There is a third way of sending one more message that is called the piggybacked response. Folks, this is amazing. I will tell you why. In the piggybacked response, it is

nothing but a CoAP acknowledgement carrying the data. This is the beauty, okay. A piggybacked response is included right in a CoAP acknowledgement that is sent to the acknowledgement receipt of the request for this response.

You have asked for something. I am giving you back without saying that I received your packet. I will not only say I received your packet, but I will also put the data back to you. Okay. Let us try sketching it as we go along. Thank you very much.