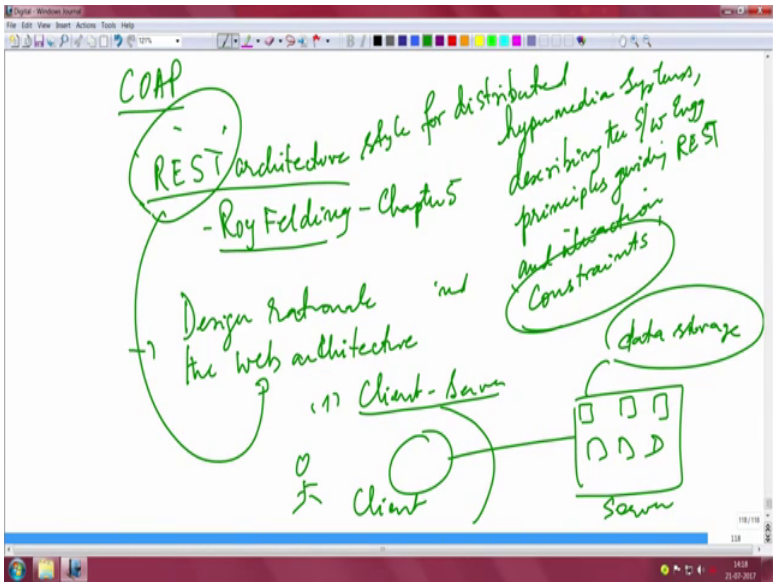
Design for Internet of Things
Prof. T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science, Bangalore

Lecture - 24
Implementation of COAP and MDNS

So, let us move on with another protocol which is called the Constraint Application Protocol.

(Refer Slide Time: 00:21)



It is called COAP, but before going into COAP what is important is to understand what is known as rest architecture and what is rest architecture? Well we can give a huge thesis discussion on rest architecture. In fact, thesis from Roy fielding the word first appeared in the PhD thesis of Roy fielding are several years ago, where the rest architecture is actually described. We will not get into that detail of I start with chapter 5; chapter 5 of the Roy fielding you should be able to find it on the web actually talks about the restful architecture, but if you want and so, there are several types of architectures for basically it is an architectural style as Roy fielding concept for distributed hyper media systems comma basically describing the software engineering principles guiding. I would say guiding the rest architecture and interaction and constraints chosen to retain those principles. So, what are those constraints under which this restful architecture for it is

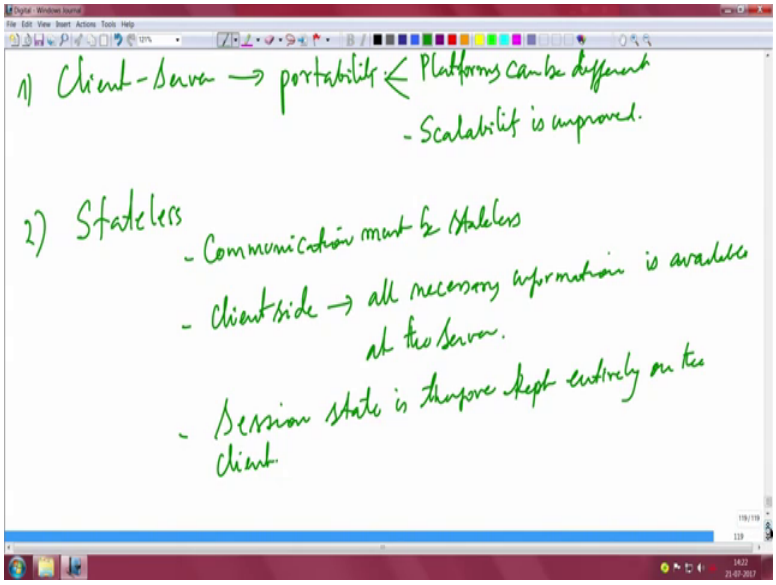
basically being a style for distributed hypermedia systems, what are those main constraints under which this restful architecture actually works right.

So, that is what you want to know. So, you would basically want to know the design rational behind the web architecture. Now you can see that I started with rest and I moved over to say I moved I started with rest and moved over to say something about the web architecture, clearly the existing web that we know very well uses this rest architecture and therefore, it is important to know some of the important highlights of this restful architecture in terms of it is constraints right. So, let me put down a few constraints and that you give your overall view of the restful architecture. First constraint is it should be client server, which means there will be a client and there will be a server holding onto some resources. Essentially this is the nice thing when we say about the style the hybrid style of the restful architecture; this is one of the important constraints for this style to work.

Separation of concerns is the principle behind the client server. So, there how do you ensure that this is a client server all right, but separation of the concert the things of interest has to be properly define otherwise you will not be able to get hold of these resources in a uniform way right. So, basically you want to separate the user interface concerns which are out here from data storage. Storage of that resource you do not want the user to be back down by where that storage is where the actual storage of that particular resource you are not really worried about.

So, basically which this clearly indicates that by doing this constraint by applying this constant of client server, where data storage and the concern for the user, the user interface concern for data storage if you d link them you basically improve the portability.

(Refer Slide Time: 05:56)

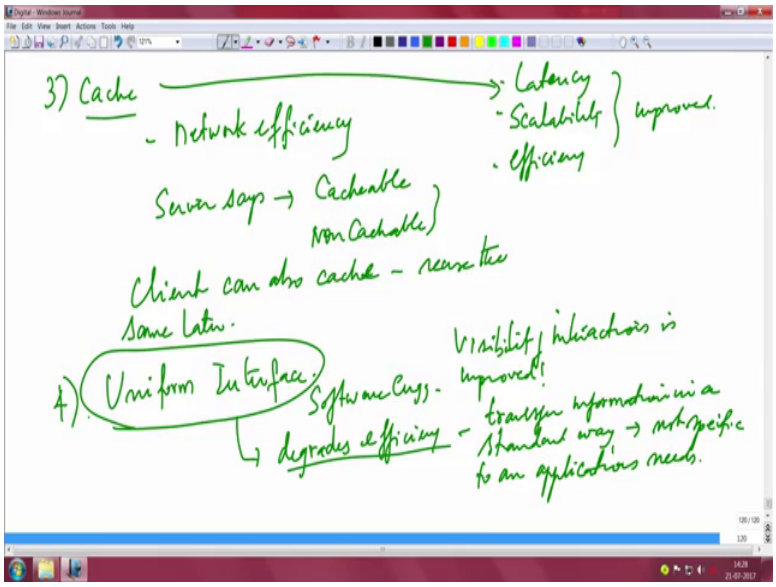


So, clients are. So, I will just put down; client server means this constraint means portability becomes automatic right. Basically when you say portability the because the user interface is independent of the client running the user interface is independent of where the storage is, you can have different types of platforms under which users can access can request for access for resources from the server side and. So, platforms can be different than you basically improve scalability right improve scalability is improved. So, this is a very important requirement right. So, separation of is not it is a key thing. Second thing is statelessness I will simply say stateless. Essentially again it goes back to when you talk about client server interaction communication must be stateless in nature such that the request from the client must contain all information necessary to understand the request; that means, it should be a self contained request which are client should make.

So, all necessary information so far from client side you should package it in a manner, that all necessary information is available where at the server it is available at the server. so that it can understand the request, essentially that is the key thing here. So, you are basically ensuring that the session state is therefore, kept entirely on the client side. So, the session state session is therefore, kept you will see all these actually happening without your own knowledge you would have actually experience many of these things when you were actually browsing; without realizing these were the basic constraints under which the web was actually working right. So, that is the next important thing statelessness ok.

What is another one? So, let us move on this thing.

(Refer Slide Time: 09:42)



The third one is cash now what about that? Basically you want to ensure that this restful architecture is also providing you network efficiency right. So, this cash constraint requires that the data within the response to a request is implicitly or explicitly labeled as cacheable or non cacheable. Server is now saying server says cacheable non cacheable what a beautiful thing.

If content does not change over time and the service says do not come back to me just to ask the value of that content, then it can simply say systems in the middle can cache it and instead of coming and the request landing on the server back it can land on any of the cached servers. So, that that information that resource can be served back to the requesting client right from that cached system right. So, that is a big advantage of the restful architecture.

If a responsible is cacheable, then a client cache is given the rights to reuse the response at any time. So, you can the client, client can also can also cache it need not be a server in the middle can also cash it, and also cache and reuse the same later that is the nice thing about this thing being cacheable. So, the advantage of adding cache constraint is that they have potentially potential to partially or completely eliminates some interactions, it will improve efficiency is scalability, user pursuit performance reduces latency. So, many things right latency improvements, latency scalability you can keep

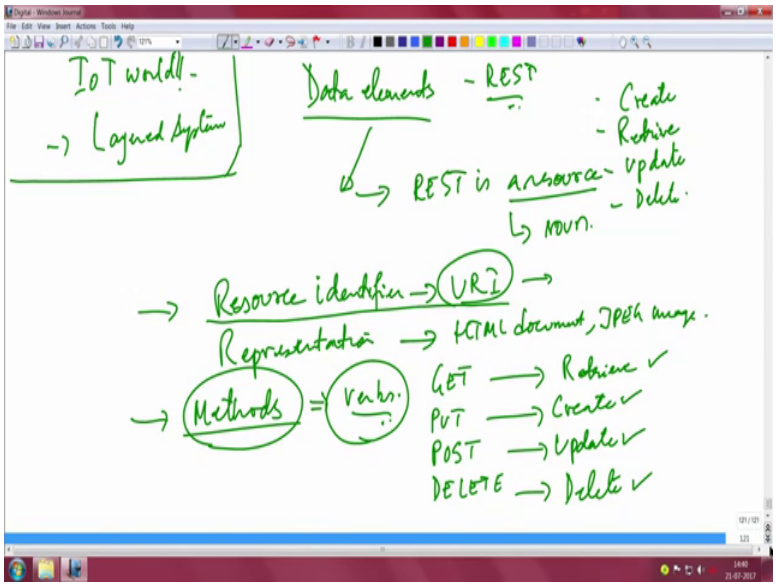
adding the features right. User sees it then efficiency are all improved right, that is the main advantage of something being cacheable the resource being cacheable. So, another thing which is perhaps most critical for you to understand COAP protocol well is the uniform interface ok.

So, when we say uniform interface, basically what you are saying is you are applying certain software engineering principles, principles which will sort of give you an interface where the overall system architecture is simplified and the visibility of interactions is improved. So, I will say visibility, visibility of interactions is improved that is the key. Implementations are decoupled from the service they provide which encourages independent evolvability basically. The trade off is that the uniform interface degrades efficiency.

So, one problem is if you talk about this it degrades efficiency, there is one every feature that you had will also have something which will create some in efficiency in the overall designs. So, really if you talk about a uniform interface, you are actually talking about some degradation in the efficiency, and why because information is transferred in a standard way right you transfer information in a standard way right. You in a standard way not really optimize for how the application wants it that is the problem.

So, not specific to an applications needs right. So, an application has to do something extra in order to turn it around to use that basic information that is available. It appears a little strange, but this is perhaps a very important constraint which you can look up and see how can I do this differently, if it is an IoT world.

(Refer Slide Time: 15:52)



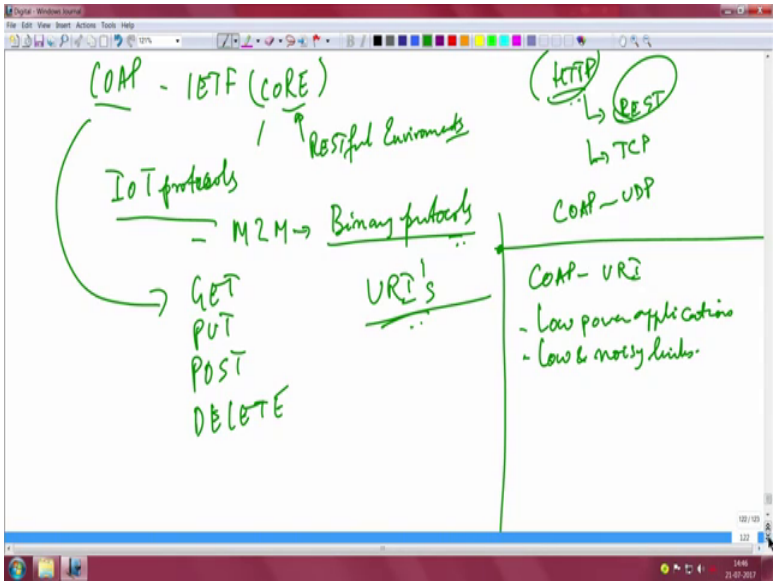
How do I do this right and that brings us to a very important point indeed. So, the restful architecture interface is designed to efficient for large green hyper media data transfer, optimizing for the common case of the web, but resulting in an interface that is not optimal for other forms of architectural interaction. So, that is what it appears with this uniform interface requirement.

You can talk about a resource identifier and you essentially talk about a URI which is nothing but the uniform resource identifier. The other thing that you should know or what are known as methods in restful architecture. So, one is resource identifier the other is the methods. When you say methods these are basically verbs and so, you have nouns and verbs that is the key point. When you say you can have verbs like get you can have put, you can have post right, you can have delete.

Let us quickly summarizes: these are the methods, methods are get put post and delete get basically corresponds to retrieve, put corresponds to create, post corresponds to update and delete obviously, means delete. So, these are the verbs and these are the actual actions that happen. When you retrieve a resource you can act on that resource, acting on that resources is through these verbs and identifying that resource through an address or through a the name noun that we mentioned basically a noun resource is basically like a noun that itself will be given a URI. So, you can act on that URI using these verbs.

So, essentially that is what would happen in the also in the in the restful architecture. Let us now use that as a basic understanding and get into the details of COAP.

(Refer Slide Time: 18:39)



COAP is essentially was also defined by IETF, and what actually came out was it was called the core group essentially it stands for constrained restful, this is r e is for restful environments. So, re is restful environments. You can see that some of the problems that rest architecture did was it did not present things to the application the way it wanted, because it had to do it in a particular format. We referred to that format in which you did let us quickly look at that from the problem of uniform interface right, it basically degrading performance.

However, if you look at this whole discussion on the IoT protocols, if you look at this discussion you are not worried about humans reading the content. You are looking at machines talking to machines machine type of communication, machine to machine communication that is why you can make it very cryptic, because machines can actually understand that cryptic language, that is why IoT protocols are binary driven they are binary protocols, which you cannot actually make sense out of if you just you know sort of unlike http which is a lot of text based, this is a binary protocol really and it can be made very very compact because of that nature. COAP actually uses restful architecture and COAP also looks at this standard verbs that we have and. So, which means there are

the usual get put post and delete correct. And there are URI s there are URI s which basically are nothing but the resource identified by these URI s.

So, you are we are trying to connect COAP with the restful architecture and that is the reason why we spent a little time trying to understand the rest architecture so that it will give you a feel. No where I have in my discussion anywhere said that COAP is a replacement for http I never said that, but you will see some peculiar, but you will see a connection because http protocol also uses restful architecture, only perhaps major difference between http and COAP is that http uses TCP and COAP uses UDP right it uses UDP.

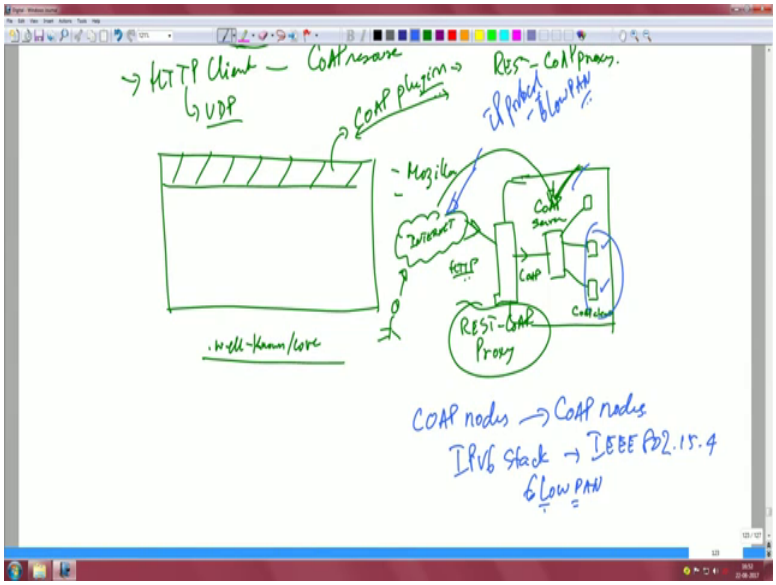
Both of them are based on the restful architecture; however, COAP is based on the constrained restful environment, it is meant for the constrained restful environments which means the header sizes are small, the packets is a small, the messages are small and less verbose in that nature right. So, that is key thing, but all the other related thing all the other related constraints which are there are very much applicable also to COAP protocol.

I will not get into the detail for instance you can look back and see that it is stateless, it is cacheable all those constraints that we applied are very much applicable also to COAP and that is where it is important for you to read and understand Roy fielding this is chapter 5 so that those constraints you revisit them properly, understand them in thorough detail and then connect it back from that abstraction of rests to what COAP can actually achieve for you right so that is the key thing. And this is an IP protocol so, which we will have to see how this protocol actually works. So, there are uniform resource identifiers COAP also means there are uniform resource identifiers and as I mentioned all the remaining verbs which are associated with that.

So, COAP is meant for low power low power applications, they are meant for low power applications and power consumption is very good if it is for low and noisy links they are meant for low and noisy link. So, that is the key thing. However, there is the one advantage of using connecting COAP to the internet. As you can see http by and large you know is a protocol which the internet understands application layer protocol, and http is well understood there are proxy proxies which seem to pass data across for example, for my private IP address system to a public address you passed through proxy

and http ports are well understood the proxies are well understood because everything rests on the restful architecture.

(Refer Slide Time: 24:40)



So, which means COAP can actually inter work and inter I would say exist COAP can exist with very small are almost no modification with proxies, with http proxies which means a very important thing. A http client can actually talk to a COAP resource a http client can actually talk to a COAP resource. Except that the http client should have some sort of because the protocol http protocol which uses these verbs is fixed accept that it should now start using UDP instead of using TCP.

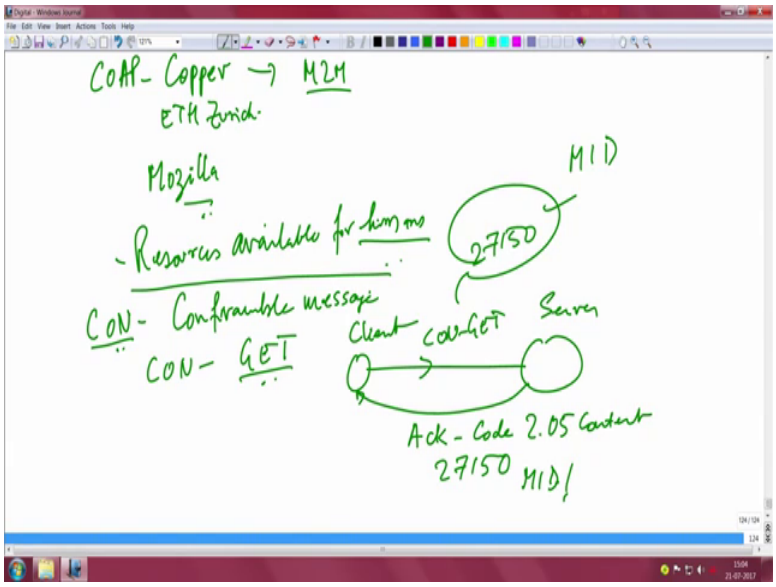
Now, if you take a browser a standard browser like it could be Mozilla or Chrome or anyone our Internet Explorer and so on if there is a nice little browser plug-in which will allow you to use UDP, but use all the nice feature of the. So, I call this UDP plug-in either I will call it COAP plug-in into the browser then all features that we are so familiar with should also work. So, that is the nice goal. Is there such a system in place can we see something of that nature well we will come to that.

Let us complete the discussion on proxies. Proxy is you can have rest COAP proxy you can actually have this and this is quite straight forward, and there is another interesting thing which means you can actually do this, you can have the internet cloud this is the internet cloud and you can have a rest based COAP proxy and here you can have complete system working on COAP you can have the COAP server here. You can have

the COAP server here and you can have all the small little devices which run COAP these are all COAP clients. So, you can see this can be http, this can be COAP all right. So, such possibilities very nicely exist and this is a proxy, which does this nice protocol translation between http and COAP.

Now, before we get into the detail I would like to point you to an very interesting plug-in for COAP.

(Refer Slide Time: 28:17)

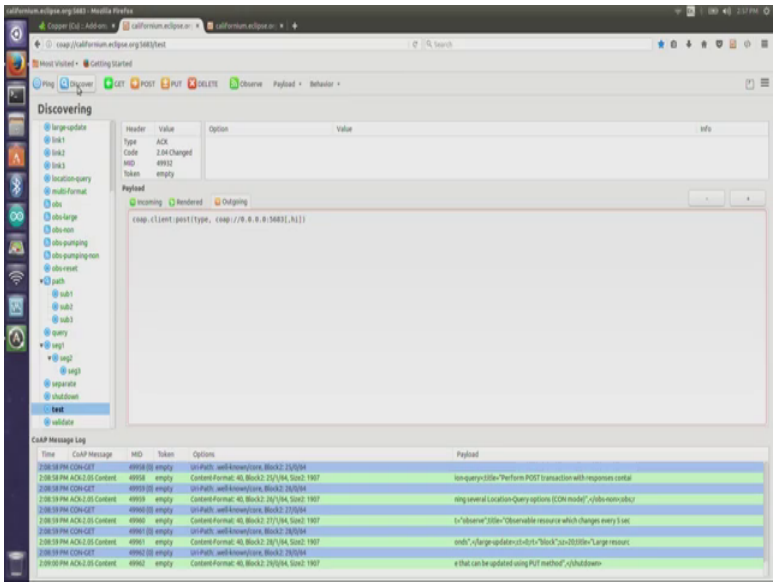


And this plug-in is called COAP interaction with copper it is called copper. Now this plug-in is from ETH Zurich built specifically for Mozilla. There is a demo there is a paper which is a two page demo paper which you can look up, essentially the introduce copper if this is basically a generic browser for the IoT COAP based protocol, because the idea is that they want to see a number of these embedded devices which are likely to appear on the internet those billions of devices, which you want to see look at what are the resources you want to look up resources offered or available I would say not offered.

Resources available for humans this is mostly to cater to humor requirements for humans, we mentioned that COAP is not really meant for humans it is between 2 devices which are IoT devices they can just exchange information between 2 systems. But if you want a human in the loop, then this copper becomes a interesting plug-in for the people to look at resources which are available on the on the web.

Let us run through a small demonstration of this system and then revisit COAP in much more detail, understand it is features of the protocol and then put back that standard problem which we often come up with which protocol to choose, whether it should be MQTT or whether it should be COAP again you will be confronted with the same to you same question you will have to make a informed choice by understanding both the protocols in extremely well in detail.

(Refer Slide Time: 30:46)



So, now, let us shift to a screen where copper is actually shown here, my 2 project staff Abhirami and Tejaswini are here. So, we will basically go through resources discovery of these of a server which is let us let me put back this picture for you. Let us say that we are here we are here on the internet, and we want to access a COAP server, we are not sure whether there is a rest COAP proxy or not it is not our concern, it maybe there may not be there it is hidden from us, let us see whether we can check the resources of this COAP server right.

Now, what you see on the screen are the standard verbs including what is known as a discover button. You can see the discover button there is a get verb, there is a post verb, there is a put verb, there is a delete verb and there is a very interesting verb which is indeed for the constrain environment and this is called the observe. We will come to that when we discuss the protocol in detail, but for the moment, let us do this discover of resources perhaps that will be the most interesting thing alright.

So, when you say discover resource you will see the whole a lot of resources which this server actually has something to offer to us what is the. So, go carefully now you look at the address bar, you will see that it is COAP colon double forward double slash, californium dot eclipse org then you have the port identification which is 5 6 8 3, slash test what we have done now is to press this is discover, and you will see that a set of resources are actually discovered.

I want to draw your attention to this first very first line, which says you URI colon dot well dash known slash core. This is a well known resource it saying something we will this will we will talk about this soon, but what is important is for you to look at this line. URI dash path dot well known slash core same this is one type of resource is telling you something, then you see the third line it is the same thing. URI dash path dot well known slash core. So, this comes back again this time the resource it is not the same resource because you have you will see that it is blocked to colon 26 dash 164 as against the previous one which was block to colon 25 slash 1, slash 64 and all that.

So, these are all the resources which the server has for us which server well it should go back to tell you that this screen that we draw here. Actually this is the server which has all the resources and you must definitely remember this dot well known let me write it here and I want you to fast forward. So, that you actually know this is a very critical thing slash core c o r e. I am not going to tell you right now, but as we go along you will see why this is a very important resource the way to discover resources.

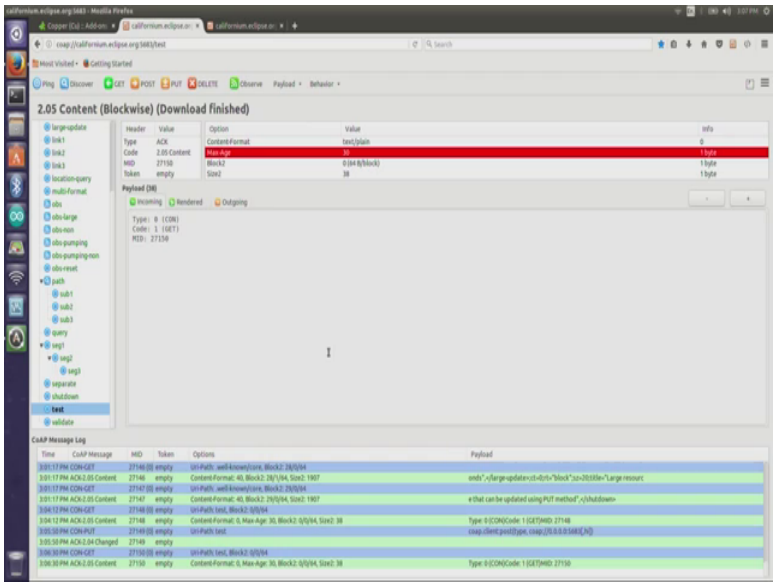
So, when you say discover you will see this nice thing then of course, the that is can get what actually happened was it did con get then you have Ack right. So, these are typical of the COAP message which is essentially going out, you will see when you say con you are actually saying confirmable right. And the verb that you are using is get that is actually that is what it means, and Ack is another type of message which is essentially acknowledge right. So, that is another thing and you if you have Ack dash 2.05 you will essentially see that that is the code associated with that.

So, if you start looking at you start a little bit fast forwarding on this front, you will realize that there is con which I said is conformable message confirmable mobile message the client is saying I am issuing you with con and I need to get something that is important this is the verb of interest. If it is a conformable message the client pushing it

out this is a server, this is con get going in this direction you will get Ack very very important and it will have a code and again you must look up this thing it is very interesting code 2.05 content, you will see it like this right good. We do not going to the detail, but at this stage it is interesting to see what actually how exactly this things are happening with respect to discover.

Now, let us go and do a put all right when you say put and we click on outgoing, your actually putting hi there just to show you that, and again you can see that it appears there in the logs in the COAP message log you will see that, and again the type here is to be noted and. So, and the message that is actually given out here is hai message. Then let us see is there anything we can do with get let us try once again, let us try a get again. So, you see type is 0 con, then you have code is one which is get. So, you can see that really you are not typing out 3 characters g e t you are not really typing any of that, you are actually giving a code called one clearly you solve binary right. So, that is the point I am trying to say it is not for humans to read these messages anymore it is for machines.

(Refer Slide Time: 38:23)



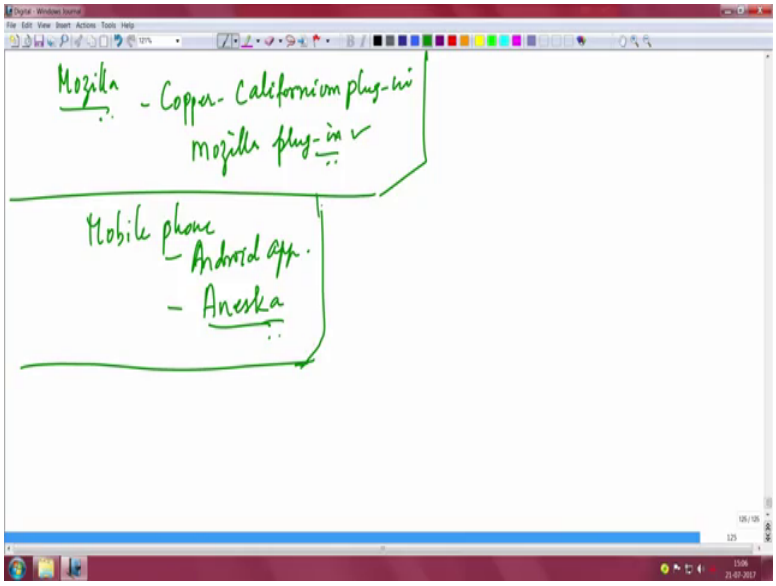
To interpret therefore, you use only this language is completely binary messages are all binary and it is clearly indicated there. So, there is indeed a there is indeed. So, you have con you have get then the message ID is 2 7 1 4 6, in the forward direction the message ID comes back with 2 7 1 4 6 corresponding to the message I d that was sent out; no message ID is 2 7 sorry it is 27105 that is the last one there. So, does not matter we can it

is actually the one that you see in the big screen here corresponds to the last message. A con get 27105 returns.

So, con get 27105 message ID comes back as in the last very last line that you see is Ack 205 dash to 05 content, 27105. Just put back to the screen for a moment you see this is important, this is the message ID you send it with a message ID the server returns it back with the same message ID clearly indicating this forward backward interaction. So, these are the 2 important things these 3 important things, discover to get and post you may also want to try.

So, how will you do all of what I have shown on screen it is pretty straight forward, what you do is go to your laptop or computer and if install Mozilla.

(Refer Slide Time: 40:27)

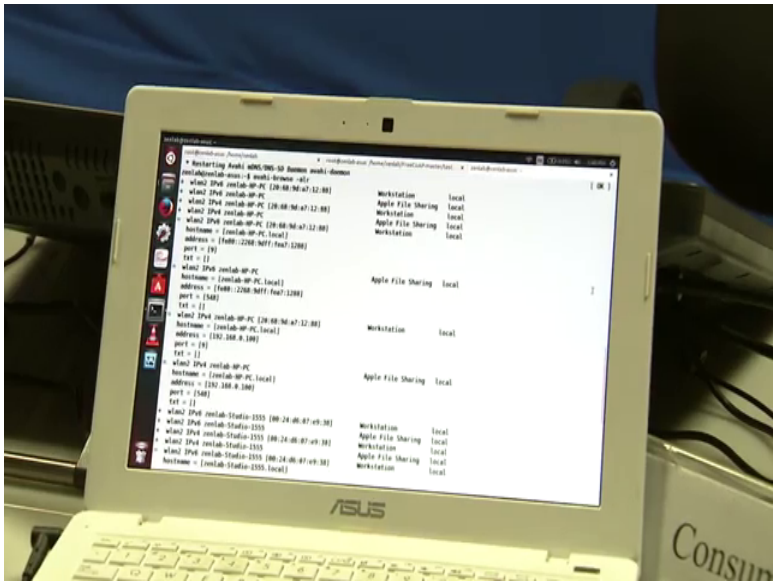


Mozilla on your system and then type for copper californium right plug-in or you can say Mozilla plug-ins you can say, Google for Mozilla plug-in and then straight away you can start working on it right. So, it is very is pretty forward.

I am sure apart from all the laptop and computers that you have it are possible that you also have a mobile phone with you. In which case you can download an android app we played around this app and this app is called Aneska. And you can also use this Aneska app to connect to standard COAP servers browse them, discover resources basically first. Before you start using any resource you should discover resources what does it mean?

Well, resources means there is a server it offers you temperature, it offers you temperature data, pressure data humidity data all kinds of information that is available is like resource, those resources you are looking for it will tell you what are all the standard resources that it has which you can use for building your application right all these things can be put together. As we go along we will see a little more in detail on this very exciting protocol called Cohap understand it to it is completeness, and then move on with other activities related to low power wide area networks and so on thank you very much.

(Refer Slide Time: 42:25)



So, this is essentially open source multicast domain name system naming system, it is called MDNS this is like a normal DNS which you know very well on the internet, but this is for IoT networks that you establish amongst your devices, gateway, h devices and so on. You do not want to really you know hardcode anything in terms of IP addresses right. So, for example, in the case of MQTT or AMQP you do not want to by heart the IP address of the broker or exchange for that matter because if it is coming through a DHCP the IP address is going to change. So, hard coding this will be an issue, as a result it is always good to have names associated with it, and somehow names to addresses can be map dynamically and that is Avahi does for you which is nothing but an open source MDNS implementation.

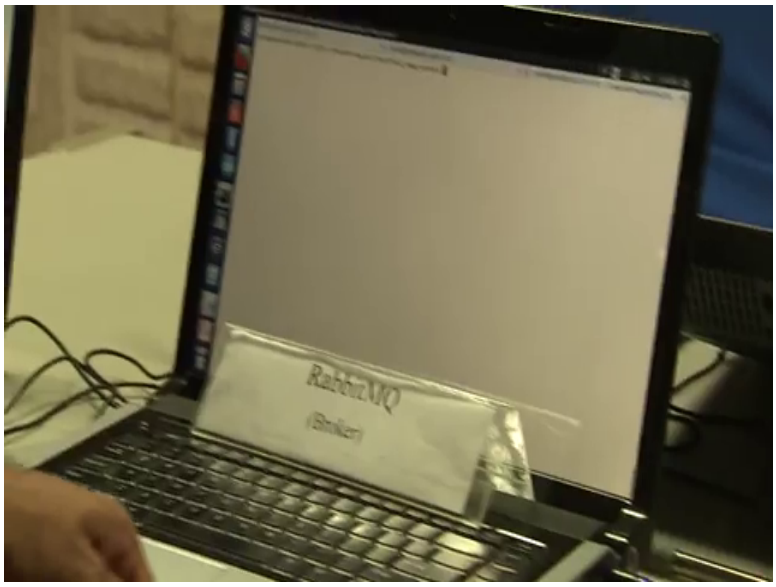
So, it runs beautifully on gateway devices such as a raspberry pi and so on for demonstration purposes we are put it on a laptop, you can see that miss Abhirami will

start this demon now and when she is starts the demon. So, it is now restarting abhi MDNS demon all right, and now what should happen is if you now in give a browse command you will see all the devices on your IoT network. Let us start actually in this IoT network what we have is 1 2 3 and 4 right these are the 4 devices which are there on our for an our this is the fourth one.

(Refer Slide Time: 44:17)

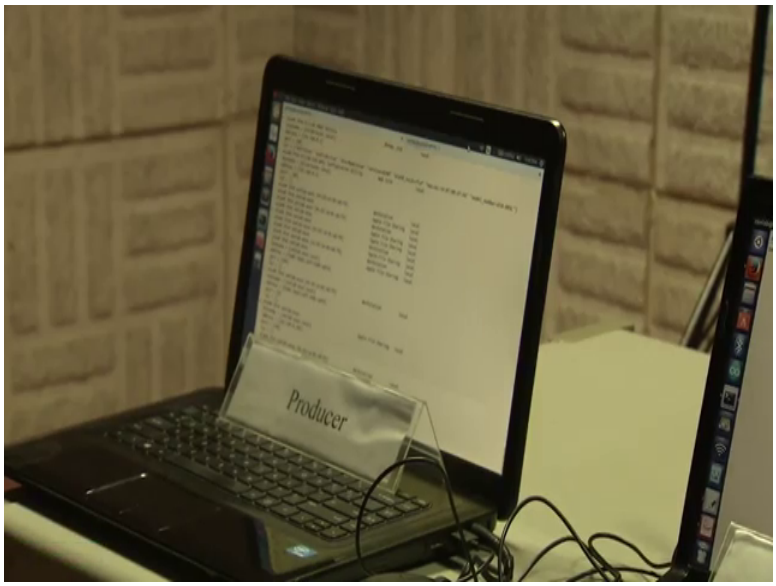
This is the third one, this is the second one.

(Refer Slide Time: 44:22)



And this is the first one.

(Refer Slide Time: 44:24)



You can see that these are the 4 devices you can say which are part of your IoT network. Are the IP address is obviously, coming from this access point gateway hardware device, which is giving out the 192.168 1. x IP addresses to all these hosts and you really do not know the IP address of either the broker in the event of MQTT or you do not know to which AMQP COAP server if you are using COAP you do not even know which COAP server, what is IP address of the COAP server of your system and you want devices to just connect and get data.

So, that is actually now beautifully displayed on the screen now let us see what are the information that is getting up getting displayed on the screen you will see that zenlab hp pc pc is this first one here right Tejaswini is also here she points that the first one that you see here the line here that you see the line that you see here essentially is corresponding to this pc here and it is actually getting displayed never mind that detail and it is actually discovering and telling you that this is the IP address, there it is giving you the name.

So, you do not have to worry so much about the IP address it also tells you the link local IPv 6 address f e 8 0 colon colon you see this is the 128 bit IPv 6 address that you can it is it is actually printing. So, that is already very good it is also telling you the. So, it is giving you in IPv 4 as well as IPv 6 at names along with the host name and address as you can see that are displayed here. The other information that is related to IPv 6 zenlab

hp pc the second line, corresponds to this no this is what this is hp this the same one right this is IPv 6 related of that first host and this is IPv 4.

IPv 4 related information yes sorry this is IPv 4 what is the IPv 4 address of the first host you can see here it is 192 168 0.100 and it is a same zenlab hp pc local which you have seen in the previous line. Now we can move on to the second host that it shows here and the second host has zenlab studio one triple 5 which is this host here, and IPv 4 address is mentioned IPv 6 address is also mentioned you can see that this is the IPv 6 address, and this is the IPv 4 address of this host. So, the headache of remembering IPv 4 or IPv 6 addresses is removed because MDNS is running on the systems and all you need to worry about his zenlabs studio dash one triple 5 dot local that is all you need to remember and all MDNS names; names on the MDNS systems will actually end with this dot local. So, that is the nice thing.

The third device is indeed itself this device here which is nothing but the IP address cause no the third device which is showing is the router here, you can see it is the d link router ideas advertising a service which is also displayed and another proprietary service called h nap which is not important at this stage, but it is a another service which is being advertised. It is called I think home network administration protocol which is a service offered directly by this system. How is all this magic happening because we did not run Avahi here on this Avahi is it that this access point is actually responding to queries by this generated by this host, simple in the case of IPv 4 it uses the multicast address to 2 to 4 and then in the case of IPv 6 it is FF02 multiply starting with f f 0 2. Moment these devices see that address coming up on the network it is the duty, if they are having services if they are able to offer services they should respond to these multicast packets and that is a nice thing.

So, you actually have an IoT based multicast DNS systems which can be put for very large networks you do not have to really worry about remembering names IP addresses of systems, but indeed you only have to worry about names. See this also gives you the flexibility of buying IoT devices from different vendors, vendors having the appropriator code and telling you that that is the name of the host that they are going to advertise right. So, that becomes very simple for integration and seamless working of IoT devices from different vendors. So, any hard coding with respect to IPv 4 and IPv 6 addresses is now eliminated completely.

Now, let us move on to a demonstration of how we can use this MDNS utility Avahi open source MDNS utility, by actually showing you a demonstration of the fact that we will show start with we will start with AMQP where rabbit and m q is the broker recall the previous demonstration, where we had to actually type in the IP address of the broker in order to either produce other the to are in order to push you are produced data, or to get your data how to consume data you had to connect with an IP address.

Now that part is eliminated by simply connecting to if you zoom in here a little bit here, you will see that you are actually connecting to as miss Tejaswini shows pica was the client for amqp pica blocking connection pica connection parameters, it is not remembering anything with respect to IP address it is Zenlabs studio one triple 5 dot local. So, you can see you really do not need to worry about in any IP address here, and the rest is taken care automatically.

And once she does that this packet automatically appear here which can also be seen that let us see if she pushes the packet, she should get a package here which is did it send a packet here on this oh it got delivered automatically is it, it is not here no can you send one more just send once stop that it is better we see if it is able to capture a nose q the message that is sent by the system here. I see only 3 what is this, this is one this is one.

Ok; so now, sorry this one now. So, now, you can see that it is nicely using the name and then pushing it to this broker, and now this system is actually hanging is actually holding on to that message. Here is a consumer which essentially is trying to get this message and it is also not interested in knowing the IP address of the broker, but indeed the name of the broker. And for that she will execute Abhirami will execute the script which says python receive underscore default dot p y, and once she gives that it connects automatically to rabbit m q broker and get some message and as you can see this here becomes 0.

So, that is a nice demonstration of how powerful MDNS is in trying to you know only hold onto names rather than worry about addresses. Let us also show you how this works with the COAP, for that again you want to connect to a COAP server and. So, let us start with running the COAP server here. So, as you can see now Tejaswini will start this dot test COAP server, which is here I will remove this so that you see it carefully it is from free COAP master. So, she runs that and now it is listening. So, the COAP server is

listening, Avahi MDNS is also running on this, now clients are trying to use trying to get a trying to connect to this server using standard conformable and non consumable messages, in this case perhaps it is trying to connect to using conformable message. So, it says conformable get is the method that is being used in order to connect to this server.

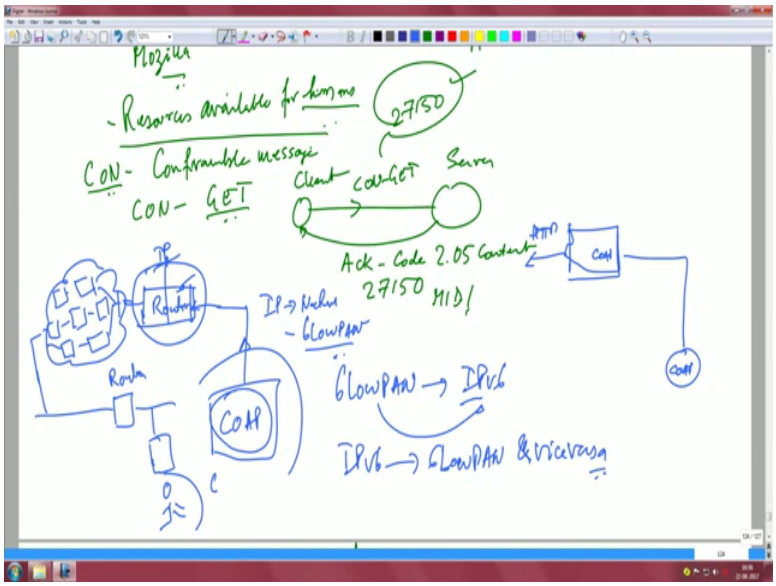
So, we will go here now and connect to that and then you see that the system has displayed without worrying so much about the IP, it says that it has sending conformable request to host zenlab studio, 1 triple 5 dot local port, 1 2 4 3 6 send to host Zenlab 1 triple 5 local and port 1 2 4 3 6 expecting acknowledgement, acknowledgement timeout is initialized it has indeed received from this host and an acknowledgement for exchanged across. So, this a clear indicator that again COAP also works quite seamlessly with MDNS systems.

So, that is in brief what is possible with MDNS combined with IoT protocols which we know well including amqp as well as COAP. So, this particular thing is so, interesting here if you look carefully at this picture that COAP nodes can communicate with COAP nodes easily without a need for any gateway nicely they will work with IPv 6 stack running on them at the network layer, in other words if you are using IEEE 802.15.4 protocol, then you can have 6 low pan IPv 6 for low power personal area network we discussed this already it mention this point earlier, this can already be running and nodes can nicely communicate over IPv 6 protocol between them right.

So, if you take this node or this node and you want them to communicate with him there is never an issue and also it is quite a seamless thing, that for instance if you do not want any conversion to from COAP protocol to http and all that and you want somewhere on the internet you have IP nodes as a systems which run IP protocol stack, which understand let us say for instance they understand IP 6 low pan for instance, I mean 6 low pan in terms of let us say.

So, let me put down a picture what I am trying to say is this. So, let me take a new sheet and go ahead with you know let us capture it quickly this are all little bit ya. So, ya. So, let us take this.

(Refer Slide Time: 57:14)



Supposing you have a COAP node which wishes to communicate directly to the internet and here there is an edge rooter just let me put an edge rooter, this is a rooter here and remember this is a internet rooter internet edge rooter, and this is talking IP protocol and this guy node on this network which is they constrained which is the sensor network is directly communicating to this, this router. So, here you will have at the IP layer or nothing with network layer, you will have 6 low pan implemented. This rooter understands 6 low pans.

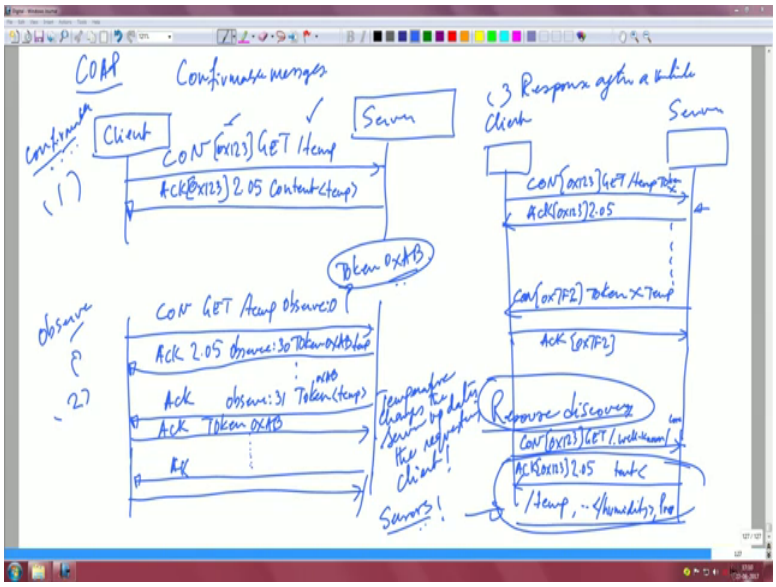
So, what he will do is, he will remove the 6 low pan part and replace the 6 low pan in terms of the normal IPv 6 and what goes out here is all the information about this COAP node except that 6 low pan got converted into IPv 6, and nicely goes into the core network of the internet. Let us say there are multiple internet routers and this edge rooter and then this attitude and this edge rooter is connected to let us say some other router and that other router is connected to the node on which there is a human sitting. And this is that remote user who somewhere on the other side of the continent who is trying to communicate with these COAP nodes and nicely it goes.

So, this route is nicely capable of converting IPv 6 to 6 low pan, 6 low pan 6 low pan and vice versa and vice versa. So, that is the beauty of COAP. So, you do not really need to have a COAP node, you do not need to have COAP node, going into a COAP conversion to http and so on. If you do not want to look at http you do not need to do any of those

proxy conversions you can directly ask this COAP node to talk to the internet. So, that is the beauty which is a clear indicator that with IoT design for IoT kind of systems where they entire protocol is running is a COAP, any remote user across the continent can also contact this COAP node directly. So, that is a very important thing.

So, that you are. So, that is the key a message take away from this particular discussion. So, this is an one important thing.

(Refer Slide Time: 60:22)



Another thing is COAP is so nice in terms of it is power, people have always said how good is COAP in comparison to MQTT and so on, but let us just put the COAP correctly into picture. If you have a client and you have a server you can do it in many many ways. One of the things this client can communicate is you can do a con conformable message with some let us say some code all right something here some number here then you should do a get right, and you do a let us say you want you are interested in temperature information. Then you get back this with an Ack you get a response code, but this has to come back this is a message ID corresponding to this con right, and response code is 2.05 and you have the accountant coming following there which is nothing but the temperature value. This is one way of doing it these are typically what are known as conformable confirm mobile messages.

Then you can also have another type of way by which this client and server can communicate, and that can be like this I do not want draw client and server here again

you can do at con conformable again you do a get and you specify what you want to get you want to get temperature, and then you q something called observe o b s e r v e sorry I do you say observe, observe then you specify what is known as. So, observe 0 call it and token to specify a token some token you put. Now this is important see the token you immediately get back an Ack with response code 2.05, that everything is fine happy and nice you get back observe with some other number and, but the token is back with the same thing with the value here temp ok.

Then time passes beauty is this is server right again you get back a message with the stake here observe as 31 is just incrementing by one, but token is the same which means whatever you started off with this token here the same token is repeating. So, you get back the same token and the new temperature. So, token is 0 x a b and the new value of temp new value of. So, let me write it clearly token is back here, and this is an Ack and for this ack so you get back another Ack.

But this time it is just taking the same token and you see as time goes by the same thing can repeat every time what is this is very powerful what does this what is this saying this is saying something very nice, every time this temperature changes the server the server updates the requesting client what a fantastic thing see again here there is a time elapse and again there is an Ack. And for that Ack again there is a Ack back from the client this will go on.

So, this can keep going nice way to remember this is how does how do systems remember this because this token is the same and every time Ack goes it always goes back with the same token and of course, the this observe can keep changing also it can become specific to whatever it is trying to do it can just give a number which is implementing, but token is very very important it is a sort of keeps the same token all the time in that way it sort of remembers. This is one way this was the previous one this was one way this was called this type one way, this is the second way is there a third way yes there is another third way which is again for lack of space I will have to redraw back the client and server and again I will put down the message sequence.

You do a con conformable some number, you do a get and then you do a get off what temperature again you have to put the token very important I will simply give it a capital X. So, that people know that we will have to use you. So, this will be Ack response code

is as usual. So, you Ack for this number right. So, you Ack for this x 1 2 3 response code has to be 2005, and you essentially do not have the temperature value here, you do not have important. So, this you observe there is no temperature value.

What the simply means is look you have ask me for a value the server is telling the Ack you asked me for a value by reading from the sensor, the sensor did not respond, but I am responding on behalf of the sensor because I am connected to the sensor and therefore, I will send you once the sensor value is ready. Then time elapses then server decides to do a con and tells the client I will put something here, but I will put back your token. So, I put back that x I am putting back your token, and now I have the temperature value. So, let me write it small so that we can fit everything into a nice picture.

So, you say con I generated my own unique message, some number and I will put back the token which is x we can put an together example, and then push the temperature value. Now I am pushing the temperature value for which you get back an Ack which is also nice right you get back an Ack. Clearly this is another way of communicating, this is type 3 and what is this type? This is called response after a while response after a while this is observe this is to is observe it should be written well right and this is conformable one is conformable ok.

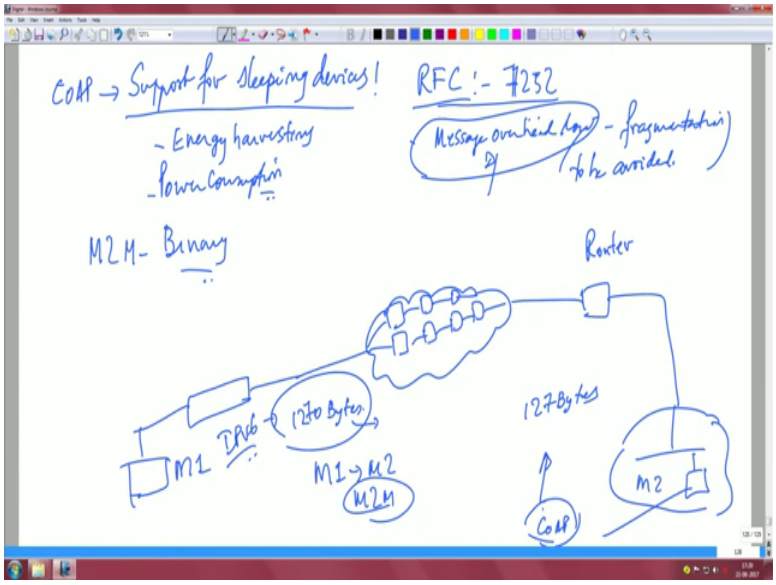
So, there is a fourth one also there is a fourth one also, which is very very critical and that is. So, you have 3 basic things which we said confirmable, observable and response after a while. The fourth one indeed is a special case it is called resource discovery and there what happens you do a con again you put back you put a number unique one do get and this time you do slash well known core, for which the server says Ack same one 0 6 1 2 3 response code 2.05, content what does it what does the content mean? You said get well known slash core whatever are the parameters that this server is sensing, all those parameters have to be are available as services right and though services have to show it could mean content open bracket.

It could mean slash temp humidity, humidity whatever are the parameters here all these parameters sensor parameters or sensors which are connected to this particular server are all highlighted in this in this Ack message in this Ack message, that is the beauty of this method and this is called resource discovery type of a message. This is very important thing this confirmable message is also called piggybacking right because you ask for get

in this case you got back the temperature almost with the Ack; that means, the temperature value requested by the client came back as a message already in the Ack and this is observed and this is response after a while and the other one is resource discovery ok.

So, this sort of gives you the power of the COAP protocol and essentially COAP is a clean design right and the design is based on rest as we said and it transfers successfully on very low links as well low bandwidth links as well and so, let us just summaries a few things which will be of great help to us right.

(Refer Slide Time: 74:10)



One of the things COAP can do an support for sleeping devices, why am I making a huge noise about this? Yes because very amenable for energy harvesting right. Remember the bearing example that we took where you wanted to split powering the sensor system and measurement of the bearing temperature, where like 2 different activities quite like the pulse human pulse measurement. One was to power the system the other was to measure the pulse of the human.

So, quite like that there can be many many situations where for saving power or for saving power, this is for support for sleeping devices why do you want to get devices to sleep because everything is related to power consumption and just to take care of power consumption you have this beautiful option of support for sleeping devices. And real time ness is maintained because you have those nice features of you know response

coming periodically ways for observe kind of messages or it could be you know something where ya. So, essentially if you are looking at real time ness.

So, if you are looking for a real time working of the system, this piggybacking works very well right. So, if you are looking at real time responses you can do piggybacking and if you want to ensure from this sensor side certain real time ness has to be maintained, after having initiated the connection and after showing interest from the client side that he has to maintain the server has to maintain values has to report values in real time if the threshold reaches, then you can use this observe right. So, you can see that without the client establishing a connection again the acts are coming periodically back.

So, this is another interesting thing and response after a while of course, this sort of while it is real time ness, but so, we do not really consider this for real time ness, but these 2 are very very good for real time operations of the system. So, this is another interesting feature, then it is not verbose this protocol is not verbose. So, which is a clear indicator this is for machine to machine communication, machine to machine communication it solve binary because there is just no point in humans talking when machines wants to talk to each other. So, I mean you do not need to put it into a way into a manner in which humans need to interpret anything. So, machine to machine communication and only point to be regarded this that this is not really a replacement for http it is not a replacement. So, do not ever consider this to be anything like a like http ok.

So, that is broadly about what the protocol is, what is very important really is to understand COAP from RFC perspective there is a full fledged RFC, it is a right document and this RFC is 7 2 5 2. I would strongly encourage you to look up 7 2 5 2 RFC and understand for any purpose of implementation looked up this RFC it will talk about the design goal to keep the message over hello overhead low, then and it will ensure that there is no need for limiting the case for fragmentation to be avoided. Remember why all these things becomes important because when you are coming from the internet world multiple routers on the internet world and one edge router here and a network here which essentially has number of this is a COAP client, here it is a case where this COAP client is directly talking to the internet world somewhere here there is an edge router and there is a machine and then there is a human here human are need not

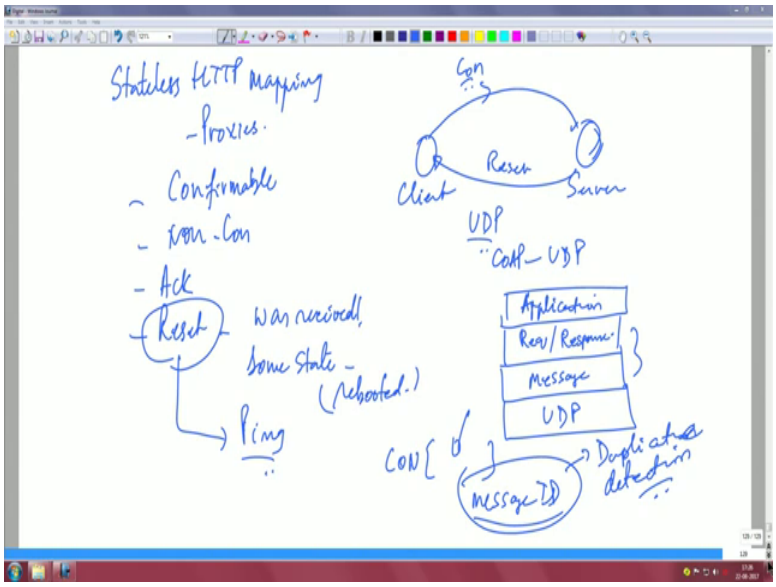
be necessarily a human it can be another machine this can be machine one, this can be machine 2, machine one is talking to machine 2 nothing but M 2 M either this case or that case.

Remember this machine one has no clue that machine 2 is a constraint not it is just talking IPv 6. So, you can have a very and the minimum mtu supported by IPv 6 is 1 2 70 files. Anyway it is more than what this node can actually take this is if you assume this to be 6 low pan and 15.4, this is usually 127 bytes. So, you must do fragmentations somewhere and send them as multiple data packets, if you are doing a seamless conversation between M 1 and M 2.

So, you do not want the header of COAP to be very big and create a large space occupancy by the header. So, the overhead of the message you want to keep it low. So, that as much as possible fragmentation can be avoided. But this problem of 1 to 7 0 minimum mtu I forget this number, clearly indicates that somewhere they should be fragmentation supported between these nodes and you do not want the overhead to eat away that space. So, keep the message overhead low.

So, this RFC 7 2 5 2 actually talks about all of that; so low overhead simple proxy, caching, stateless http, stateless http.

(Refer Slide Time: 81:43)



Http mapping, which is a clear indicator that proxies can come in nicely it integrates to the internet current internet. So, that is the beauty of these protocols, and then what else is important. So, I mention to you about the 4 types of messages which is non conformable message then acknowledgement message which is so conformable. So, let us look at it conformable non conformable, then Ack message and then reset not much has been spoken about it.

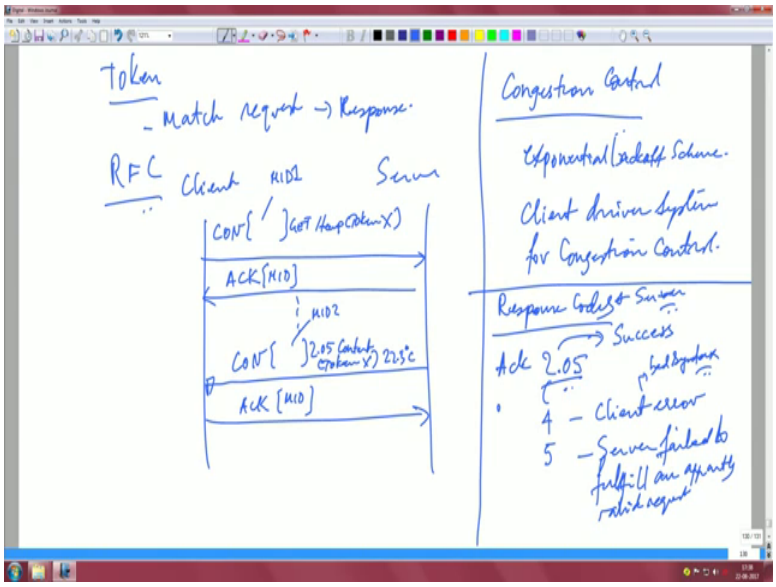
So, let me say a few things this indicate that a specific message was received you I got it, it simply says I got it no problem, but I sort of do not understand some state I sort of do not understand some state in it, perhaps because I rebooted this is what the server is saying client has send the message who message was received for some reason some state of that message was forgotten. And therefore, either the due to you know because the server was rebooted or something. So, you basically forget. So, you are saying that I received it right. So, what can you say about it is very interesting thing that even the RFC is says is that if you want it is like you can use it like a ping.

You just want to find out whether that server is alive or not. Then what you do you send an empty conformable message you send a conformable message and you say. So, and then you just see whether the server responds is I got your conformable message, but the meaning of that conformable message does not make sense to me and therefore, I am giving you a reset. You are sending a conformable this is client this is server and this is the reset. Something that you formed in the conformable message that you sent across was not suitable for the server; however, he got the message and therefore, he did a reset so that something that you can actually try ok.

So, as I already mentioned COAP uses UDP. So, that is another good thing COAP uses UDP, and the stack if you look at the way it is shown is that you have application layer and the stack of COAP can be divided into 2 parts, one part is called the request response and the other part is called the message, and down below is UDP which is the transport protocol response. So, that is interesting binary fixed length message and whenever you say I describe to you con message with some number there, this essentially is nothing but the message ID. I said con and something right here this is the message ID that is important all this is described very well in the RFC you should not even worry too much about what is shown there because that is clearly indicated in the RFC please do read the RFC in it is completeness so that many things can be understood directly from the RFC.

Message ID reset is for basically why do you need to have this message ID, this message ID is to detect duplication. So, detection duplicate detection is nicely taken care by message ID. So, that is a very important thing and we also mentioned about the token in the previous example.

(Refer Slide Time: 87:33)



In the example that when we took the token is used to match the request with response request. Response matching is done using this token that is very important and it clearly comes in the protocol stack, which we described earlier by showing that request response is indeed one part of the layer in the COAP system. So, to much this you nicely use the tokens and duplicate detection be already mentioned is done by the message ID ok.

So, that is a very very important part, if the server is not able to respond immediately to request carried out in a conformable message, you can basically the server can issue a empty con message. So, as I mentioned that is something that in the message is that we looked up that is all there. So, that is also interesting and whenever the server is ready at the sensor is switched on in all that, the server can send the new conformable message which essentially is actually called the separate response. So, that is another interesting thing there is something called a separate response, you can write this you can also. So, essentially all of this is beautifully written in the RFC.

So, I am actually reading things from the RFC for you, you should try and find out I will go through this in detail to understand between the client and the server. This is the

client, then the server, you send a con message right and you put the message ID here you put that. So, here you put message ID and you put get the work that you are interested in is get the sensor value let me write it neatly con message ID get this is the verb of interest temp. Remember all this looks very familiar to the URI that we described earlier with a token, I will simply call it x. so that we can meet him and you get back Ack with the same message ID. So, that duplicate detection is possible right and time passes we mentioned this again and again what you get is from this side you can get a con.

And essentially get back new message I d MID 1 this is, this has to be a MID 2 right and you get a response code because this is really a response from the server with the content token is whatever was used here. Just like what we described last time right token exactly the same x with the value, let us say it some degree Celsius for which the client says I got it I am acknowledging you with what when the same message ID. Because you want to ensure that in case he did not get it you will create with a new message id. So, all the duplicate detections are actually possible.

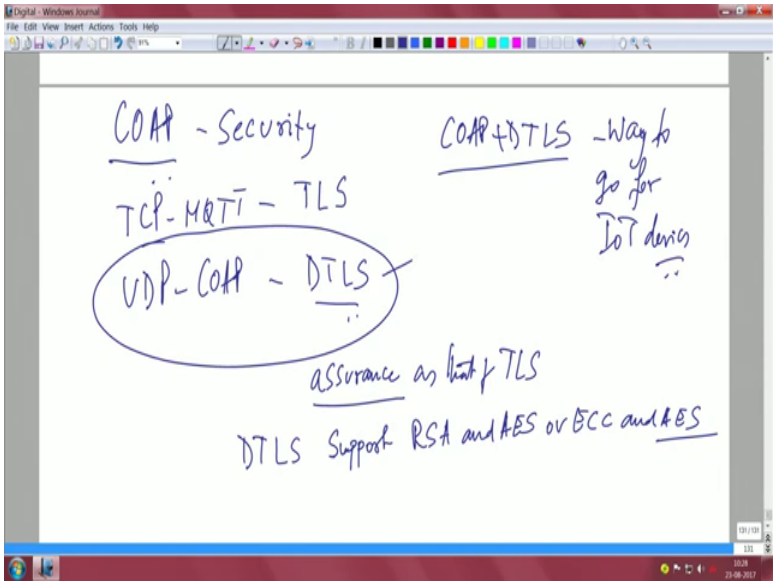
So, in summary all this is something that you can easily do if you can easily understand, if you read up the RFC. The RFC also talks about in addition to all what we discussed it also talks about condition because this is what you may encounter when you are trying to connect the COAP clients to the internet. Basic congestion control for COAP is provided for exponential back off mechanism that is already mentioned. So, there is an exponential back off scheme implemented, essentially you send a con you do not get back an ack. So, you do an exponential back off and you re try sending that message. So, essentially things like that are note, as thing is the things such as simple as these are actually done right. So, you may have to also look up that section on the RFC to look up the congestion control, but you know there is no specific algorithm which the client you know the client has to sort of; you know there is no hard and fast rule about by what is the how soon should the response to a conformable message no I will I will put it this way.

The specific algorithm by which client stops to expect a response to a comfortable request that was it acknowledge or to a non conformal request is not defined. So, this is important. So, in summary quiet places the owners of congestion control mostly on clients, so client driven system for congestion control, this is a most important thing. So, let me also tell you little bit about the response codes. Again this is from the RFC, so,

you could look this up in great detail, we saw that Ack came with 2.05. In fact, even in the demos we saw that simply this mean. So, we can have to here you can also have 4, you can also have 5. Too simply means success, 4 means client error and 5 simply means server error. Server error or you can say server failed I calls it server failed to fulfill an apparently valid request ok.

So, client error this could be because the request contains a bad syntax or cannot be fulfilled and this is nothing but a bad syntax remember when you say response codes this is from the view of the server right this is a view of the server. So, this is also clearly mentioned in the RFC. So, please I once again request you all to read the RFC in great detail so that you will be able to understand the protocol very well. Like MQTT where we pointed you the standard you should also go through this RFC in great detail, to understand about this protocol in completeness.

(Refer Slide Time: 97:41)



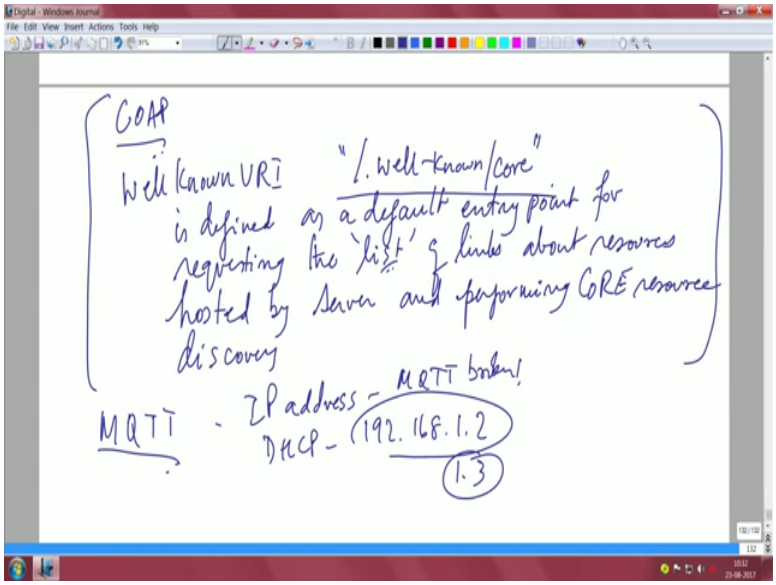
Let us wind up on COAP I must tell you that as far as COAP is concerned we did not discuss anything with respect to security right secur s c sorry s c c u security. IoT success design of IoT course big success means you may have to look at power, you may also have to look at security configuring COAP protocol is just one part of it. Configuring COAP with a secure protocol with security features enabled in it is the second most important part which has to coexist with the protocol itself. Now what does COAP offer you in terms of security. Well nothing less than what MQTT actually did, MQTT had the

advantage that it uses TCP right we are talking of UDP and COAP and this was done with TLS right. Now here it will be DTLS that is all datagram transport layer security. So, there is no compromise in security as far as COAP is concerned please note this ok.

So, COAP provide security by DTLS that is already said, DTLS is nothing but datagram transport layer security it provides the same assurance same ditto assurance as that of TLS no difference at all to TLS and the good thing is it transfers data over UDP. Typically DTLS capable COAP devices will support DTLS capable devices, will support DTLS support will support RSA and AES or is ECC very well suited for embedded applications and AES. So, please explore the RFC of COAP and get into the details of DTLS and have your DTLS protocol with security DTLS. So, COAP plus DTLS is the way to go IoT devices all right the something fantastic this is one part.

There is a fantastic part.

(Refer Slide Time: 100:38)



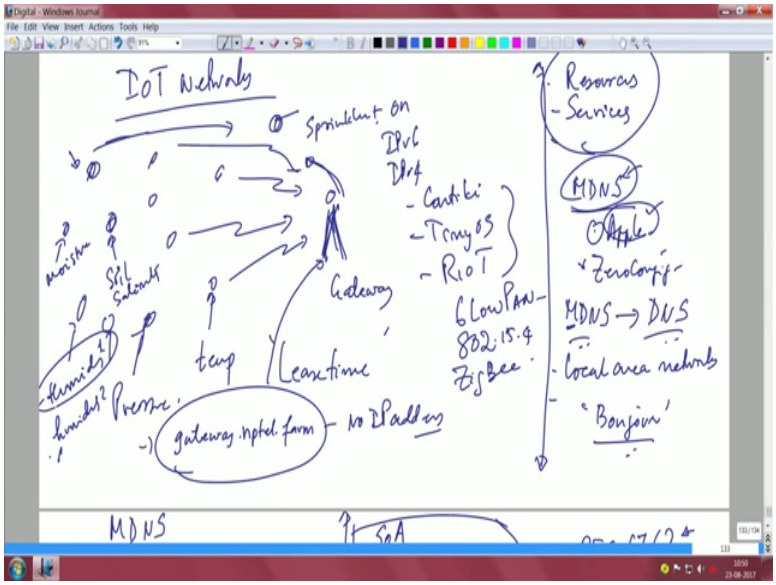
Sorry there is another fantastic part with respect to another important thing about COAP last perhaps the most important least, but last, but not the least is the well known URI what is the well known URI? It is nothing but the slash dot well known slash core, do not forget this URI. This URI is defined as a default, is defined as a default entry point for requesting the list of links about resources hosted by server and performing rest sorry core I would not say rest core anyway is constrained rest right core resource discovery. What a beautiful URI available for you which automatically is a default entry point and

you could have a list of links which essentially are nothing but resources which are hosted by the server this is a for as COAP is concerned you do not have some this kind of nice feature with MQTT right.

In fact, it is even hard to find the IP address or the MQTT broker, it is hard to find. And why is it hard? Because today you set up a IoT network with a DHCP enabled a system, you get IP address for the broker as 192.168.1.2 let us say, and tomorrow the same one will become one dot 3 you cannot go and change the source code of each of these clients in a manner that they should contact this broker with the changed IP address this is bound to happen in large IoT networks how are you going to solve this problem.

Well, here is a nice way of doing things and this is something that you should definitely consider every time you set up large IoT networks, and that comes in the form of a nice lan based for dense network based systems for dense IoT networks.

(Refer Slide Time: 104:03)



IoT networks many many many IoT nodes in an in a network, which are randomly I would not put them in line, but I would put them scattered right. So, that scatter them in all possible ways and there is a sort of a gateway node which is you know communicating this is simply the gateway node. Each one of them are offering different services take this node he may be monitoring temperature take this node he may be offering he may be monitoring pressure, this may be monitoring humidity, this may be monitoring soil salinity, this may be monitoring soil moisture and so on and so forth

many many nodes many of these nodes are offering services, and there is one node somewhere here which wants this data so that it can control something back.

For example, if the moisture content is low it may decide to switch on the sprinkler pam and this is a large IoT network in a in a field in a farm right. And every day if they are all running IPv 6 and IPv 4 it does not matter, they are all running IP stack protocol with embedded oasis such as contiki or tiny os or riot os any one of these embedded operating systems which support 6 low pan which I mention to you is on 802.15.4 and the kind of a mac file which is also called the zigbee nodes, but is it zigbee protocol if they running all of that, they should be able to contact and find out this node should be able to find out the simply get the data from these nodes and these nodes have having difficult to contact the MQTT broker. Let us say this is running MQTT there even finding finding it difficult to find MQTT broker, because the IP address of these nodes change and so does the IP address of the broker changes because the lease time the least time of the DHCP system, protocol has expired and new IP addresses have to be assigned and therefore, it gets a new IP how do you solve this.

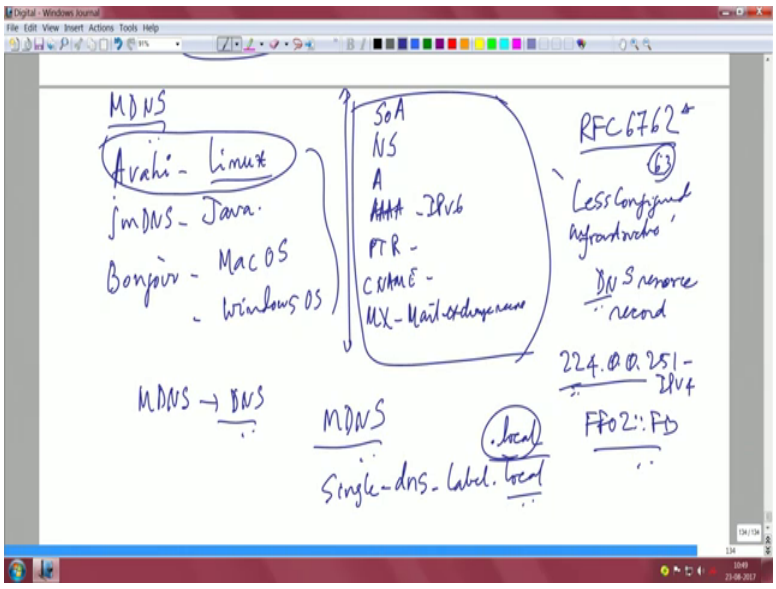
Well as I mentioned to you the nice answer to this problem in terms of finding resources discovering resources, discovering services, all of that is possible with a protocol called MDNS. And this is in indeed how apple you will be surprised apple was one of the first to implement 0 confic, they were able to find devices apple devices were able to find you know devices within themselves and they will discover devices, discover services, discover resources, Chrome cast for instance another well known application which uses the fact that they are able to discover right devices.

All this is possible because MDNS is a solution to this problem these are light weighted it is exactly the same MDNS borrows everything from the well known DNS exactly the same, but were meant for local networks local area networks. And most well suited for IoT applications where that these numbers of nodes are installed and you simply install MDNS here and let these nodes actually send out a multicast discovery packet. So, it uses multicast DNS MDNS means multicast DNS these nodes simply shootout in multicast DNS packet, and then response the systems the gateway or other nodes actually respond with the services and resources that they are available.

So, one of the simplest is you just want to find out who is your gateway for example, here you program not anymore and IP address, but you say I will just give a name gateway. Gateway dot nptel dot com. Let us say this is the gateway name gateway nptel com gateway of this nptel course this is a gateway you are not programming any IP address no IP address just configure this. Now you shootout I want to know every day if you if you want to be discovered the IP address of this and this indeed this system here is that raspberry pi or odroid or any one of these embedded systems gateway kind of boards which actually are configured for getaway dot nptel dot com. So, this is actually this and if you run MDNS on this, he is the one who will respond back for this nodes query that I am interested in knowing the IP address of gateway dot nptel dot com, if this request goes out as a multicast packet the response comes from this node by saying hi this is my IP address. So, that is a beauty of using multicast DNS over large IoT networks ok.

So, and apple in indeed is something that was one of the first to implement this 0 configure, the call this bonjour; bonjour perhaps and they call this o b o n g o URI will not spell it for you because it is nothing not native to what I speak. So, all right. So, it is also called 0 configuration or also called by this name and apple seem to implement this many years ago this is possible to enlarge IoT networks therefore, let us now look at MDNS in detail.

(Refer Slide Time: 110:53)



Spend a little time understanding MDNS. MDNS essentially you have different implementations of MDNS Avahi is an implementation on linux, and jm DNS is an implementation on java it is a jm implementation is on java and bonjour I mention to you is an implementation on Mac and bonjour is also available for windows all these of popular operating systems actually do support you know MDNS system.

So, the point that I am trying to get at is that MDNS is a very important application that you must seriously consider when you are talking about you know trying to use trying to discover services and resources on a large IoT network. You must be a familiar if you are not please look up the standard DNS records that are well known, you have start of authority S o A kind of record, NS for name server record, A for address record 4 A quad a for address for an IPv 6 world and PTR nothing c name is nothing but the canonical name also well known which is alias valid points to be canonical name of a host identified by the by an a record, then you have MX which is the mail exchange record ok.

So, these are. So, these are some things that MDNS actually will seamlessly work with DNS, that is the most nice thing. So, if you want to use your IoT network and scale it up into a larger connected to the larger internet world that is something that you can easily do with the with this same MDNS protocol. So, let me get into a little more detail of this this exciting protocol and you must note that the MDNS is actually defined in another RFC, and the RFC indicated by I just pull out the RFC is 6 7 6 2. So, please look up RFC 6 7 6 2 and going into reading that the abstract itself is.

So, beautiful it says network devices become smaller more portable and more equator, the ability to operate with less configured infrastructure is increasing is increasingly important less configured beautiful sentence, less configure infrastructure. Less configured infrastructure is increasingly important and in particular the ability to look up DNS resource record, record data types in the absence of conventional DNS server is useful. What a beautiful sentence; that means, you do not need to worry about the internet you do not need to worry about presence of DNS on the internet, but you can actually configure your own multicast DNS right there without right very simply by installing Avahi on if it is Linux based systems right.

And they it uses multicast which means you must be looking at what is an outcast address; obviously, in IPv 4 world everything starts with a multicast start with 2 2 4 and. So, if you send out a multicast packet with 2 2 4 dot 0 do 0 with the complete IP multicast IP address then there will be a response from MDNS capable servers, and it is free it is free to use that is another nice thing.

There are other companions protocols called DNS based service discovery which is 6 7 6 3, but that is a called s d it is called service discovery these are the companion technology protocol, but never mind what is important is that 6 7 6 2 is already good enough for you to follow and get a hang of how you can install MDNS on it. So, how should you assign names now you have many many of these devices right you must you. So, there are many of these devices and this gateway needs to know which one of them is actually offering soil salinity, which is offering moisture soil moisture humidity pressure temperature and so on. That means, they should also have an MDNS name associated within not just the gateway, because this sprinkler is the one this node essentially is the one that is actuating is sprinkler in case moisture is soil moisture is low that is what we said.

So, they must also be the gateway also should know from which node of the several nodes that actually, particular sensor data is being is made available therefore, there is a convention by which you should associate names MDNS names to these to this large dense IoT networks and essentially it should end with local that is all that is required. You can have any name, but it should end with local so, that is an important requirement.

So, essentially what we are saying is that a host that belongs to an organization or individual who has control over some peritoneal DNS space can be assigned a globally unique name, within that portion of the DNS namespace such as something right some example dot com or something. In fact, if you look at 6 7 6 2 it gives you the an example of example dot com for those of us who have this luxury this works well. But here you are not even talking about internet you are talking about something which is not having any access to a global DNS namespace it is local, local to your own IoT network that you are assigning names to different sensor nodes.

So, that resources and services can be discovered quite effectively and therefore, the problem is forget about internet just local to you give your computer systems, your

sensor nodes link local multicast host names of the form like this single dash DNS dot not that dash DNS the label some label dot double dot local as I mentioned that is very important. Any computer user is granted the authority to name their computer and sensor node in this way, provided that the named provided that the chosen name is not already in use on this link you are to be a bit more careful do not assign the same name to multiple nodes, but keep it unique within your network which is in your control, having named that count computer or note this way the user has authority to continue utilizing that until such time there is any conflict which again you yourself should resolve it locally. So, that is a very important thing.

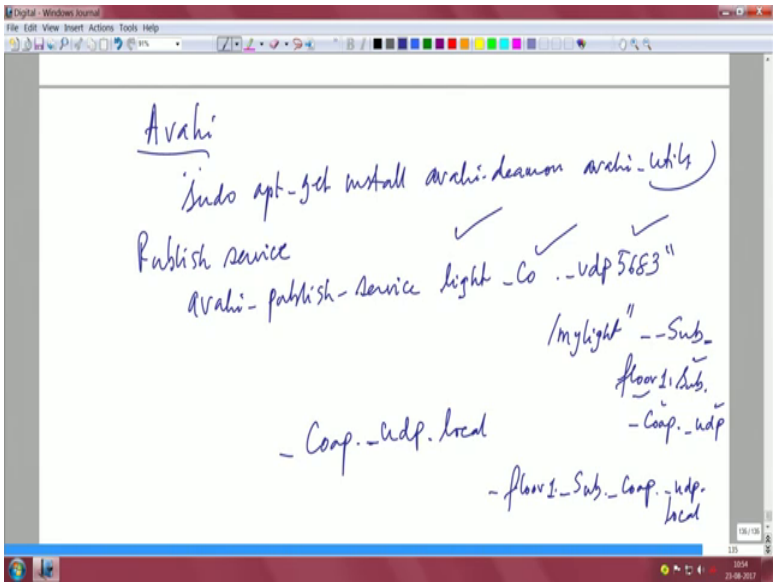
So, if it is IPv 6 world you will essentially use FF02. So, let me just complete this if it is IPv 4 world you use 224.0.0 let us say 2 5 1 this is a for IPv 4 use this and send out a multicast to send out the multicast packet and use FF02 colon colon FB as a multicast packet to discover resources and as I said you must assign a local to it. So, that you will be able to you know use this MDNS in the right proper way without violating RFC 6 7 6 2.

So, that is a very important things and query in querying of information in DNS is a many types you have one shot queries, and you have ongoing multicast queries and so on. So, really if you read that RFC you will be able to understand appreciate what the difference between one shot and continuous multicast DNS querying is all, but essentially it is this you I give you a very simple intuitive examples suppose if there are 2 nodes measuring humidity, this is humidity 1 this is humidity 2 by these 2 nodes.

Now, this actuator node should take which one of them. Now one way one simple way is to say I will take whichever responds first that is one shot the other way of saying it is that I am going to wait for multiple nodes to respond and I will take that humidity which is the highest because they are specially separated for some reason and humidity is different in these 2 locations, you may be interested in greater humidity value as compared to the other one. So, let us say humidity 2 is greater than humidity 1 if you just do one shot, and if the first node that responds is humidity one sensor node you may not want to take the decision. Therefore, very interestingly you should be able to use the DNS in it is complete set of features which are there ok.

Now, let me quickly summaries all of this by taking the example Avahi.

(Refer Slide Time: 121:51)



Awahi can be installed very simply on any raspberry pi odroid which is running Linux by simply doing the pseudo pseudo apt standard ubuntu commands get install get install Avahi Avahi demon, Avahi you until lot of this essentially is the MDNS demon along with that comes a set of tools utilities for you to do query to browse and so on. So, you can easily download this and you can publish you can use Avahi you can use the tools provided by Avahi to publish services.

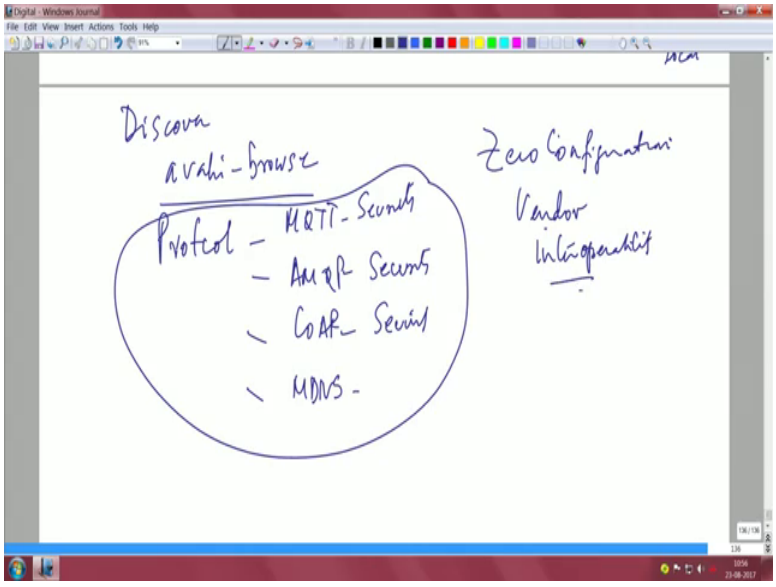
How do you publish how do you public services sure there is a way and you simply use Avahi publish service and it gets it is a service name of course, you give a service name and then it gets nicely published right. For instance you can say Avahi publish service light underscore COAP dot underscore UDP 5 6 8 3 open quote sorry this one open quote slash my light right dash dash sub underscore floor one dot sub dot slash COAP dot underscore UDP ok.

What will happen if you do this it will publish a service called light, it will publish a service called light which uses COAP protocol on port 5 6 8 3, the same service will be discoverable under COAP UDP dot local read it I will write that ya actually I shall be doing that. I should take the same service should be available under COAP dot under UDP remember local should be there local and floor one sub COAP UDP right and there should be local as well. So, I will not this is the syntax of write of doing a publish, but in understanding it should be underscore floor one dot the dot underscore sub same thing

you should write here, sub dot dash COAP dot UDP I am repeating all that is here, but I am just adding local. So, this is what it is.

If it is successful the process will remain active and it will continue to offer these beautiful services; you can discover not only published, but you can also discover services.

(Refer Slide Time: 125:53)



You can also discover service using Avahi you say Avahi browse and you will discover services browse for all service types resolve service automatically, you can display output in parable format and all of that. Discover all COAP service servers and automatically resolve them if you want to do that there is a nice syntax. Read up Avahi syntax and you will be able to understand discover the COAP server at floor one, and show it is parsable t is in a parsable format that is also possible in other words and in big summary.

If you look at the protocol world of IoT, you must know MQTT and it is security features you must look at AMQP and it is security features you must look up COAP and it is security features, you must install MDNS for resource and service discovery all of that put together will essentially ensure that you will have the luxury of 0 configuration, and very importantly vendor interoperability ability.

Suppose if you by an MQTT node which is considered which is from one vendor and he would have given a let us say a URI right and that URI essentially is let us say connecting to some a broker, broker name itself will be like it will not forget URI you just think about it is configured with a name for a broker.

Now you have another vendor giving you the broker hardware and all you have to do is just make the modifications to assign that name to it that is all. Then automatically across vendors you should be able to have a interoperable way of node sensor, node from one vendor and broker from another vendor. And still they should be able to interoperable they should interoperable. And they should be able to discover services, they should be able to discover resources and work seamlessly.

Thank you very much.