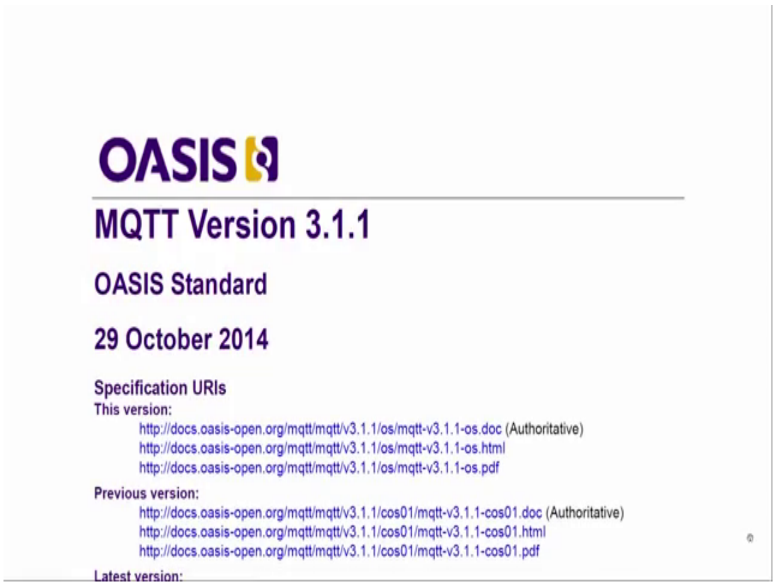
Design for Internet of Things

Prof. T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science, Bangalore

Lecture - 22
Standards and Security in MQTT

Moving on with the MQTT it is useful to know; what is actual standard with which one should implement MQTT. Suppose you want to start implementing MQTT from say you want to develop the full protocol and the relate commands, which are part of the part of the protocol you should know that right. So, the standard should be known very well that is indeed the basis for anything. Now, let me point out you point you out to the actual standard the standard is from oasis, it is called another standard which and the current version is MQTT 3.1.

(Refer Slide Time: 00:50)



OASIS

MQTT Version 3.1.1

OASIS Standard

29 October 2014

Specification URIs

This version:

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.doc (Authoritative)
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf

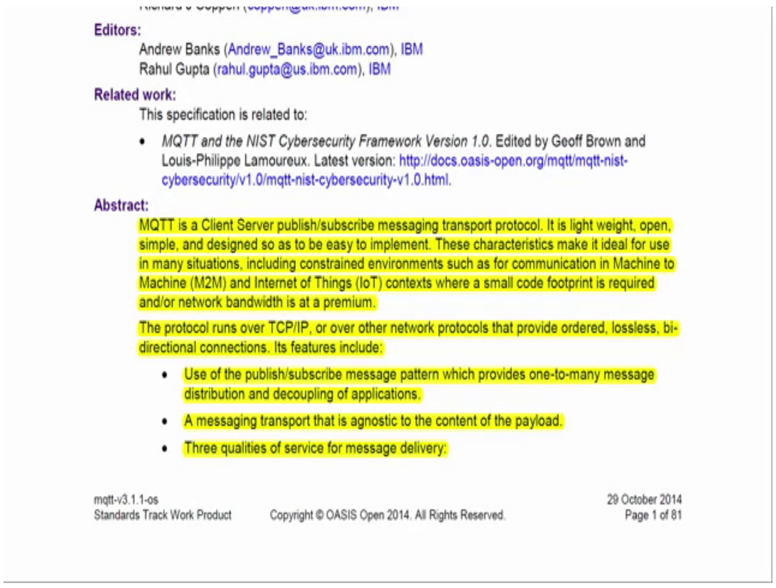
Previous version:

http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/cos01/mqtt-v3.1.1-cos01.doc (Authoritative)
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/cos01/mqtt-v3.1.1-cos01.html
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/cos01/mqtt-v3.1.1-cos01.pdf

Latest version:

If you look carefully into this standard it tells you everything about the explanation of the protocol. The abstract is I would say one of the best abstracts I have ever seen which describes the full protocol.

(Refer Slide Time: 01:07)



Richard J Coppen (coppen@uk.ibm.com), IBM

Editors:

Andrew Banks (Andrew_Banks@uk.ibm.com), IBM
Rahul Gupta (rahul.gupta@us.ibm.com), IBM

Related work:

This specification is related to:

- *MQTT and the NIST Cybersecurity Framework Version 1.0*. Edited by Geoff Brown and Louis-Philippe Lamoureux. Latest version: http://docs.oasis-open.org/mqtt/mqtt-nist-cybersecurity/v1.0/mqtt-nist-cybersecurity-v1.0.html.

Abstract:

MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium.

The protocol runs over TCP/IP, or over other network protocols that provide ordered, lossless, bi-directional connections. Its features include:

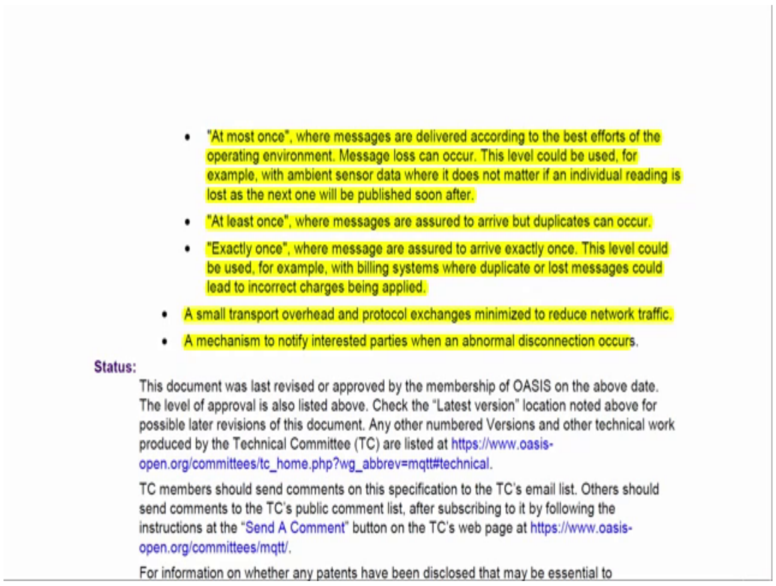
- Use of the publish/subscribe message pattern which provides one-to-many message distribution and decoupling of applications.
- A messaging transport that is agnostic to the content of the payload.
- Three qualities of service for message delivery:

mqtt-v3.1.1-os
Standards Track Work Product — Copyright © OASIS Open 2014. All Rights Reserved. — 29 October 2014 — Page 1 of 81

I just read it in fact; it is the real highlight of the thing. It is a client server publish subscribe messaging protocol it is very well written lightweight, open, simple and designed to be used by design so as to be easy to implement. These characteristic may be ideal for in many situations including constrain environment, such as communication in machine to machine and internet of things context, where a small code footprint is required and or network bandwidth is at a premium.

The protocol runs over TCP IP or other network protocols, that provide ordered lossless bidirectional connections. It features it is aid it is features include use of publish subscribe message pattern, which provides one to many message distribution and decoupling of applications. A messaging transport that is agnostic to the content of the payload, 3 qualities of service for message delivery you know this well now, we saw some demonstrations of it.

(Refer Slide Time: 02:25)



- "At most once", where messages are delivered according to the best efforts of the operating environment. Message loss can occur. This level could be used, for example, with ambient sensor data where it does not matter if an individual reading is lost as the next one will be published soon after.
- "At least once", where messages are assured to arrive but duplicates can occur.
- "Exactly once", where message are assured to arrive exactly once. This level could be used, for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied.

- A small transport overhead and protocol exchanges minimized to reduce network traffic.
- A mechanism to notify interested parties when an abnormal disconnection occurs.

Status:

This document was last revised or approved by the membership of OASIS on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment" button on the TC's web page at https://www.oasis-open.org/committees/mqtt/.

For information on whether any patents have been disclosed that may be essential to

At most once where messages are delivered according to the best efforts of the operating environment, loss can occur this level could be used for example, with ambient sensor data, where it does not matter if an individual reading is lost as the next one will be published in soon after see.

Essentially when you are measuring an ambient environment, the temperatures do not change drastically. So, even if you lose one packet it does not matter right. So, that is what it says. So, look at how it can be applied. At least once where messages are assured to arrive, but duplicates can occur there is really no harm if multiple messages containing a sensor data actually happen right. So, in that situation you do not mind even if it is an at least once kind of thing, you have a way by filtering out multiple messages that come there is intelligence on the system to remove those messages that is something that you can actually do.

Exactly once where message are assured to RF exactly once this level could be used for example, with billing systems where duplicate or lost messages could lead to incorrect charges being applied. You obviously, do not want your utility bill to produce your utility bill to appear twice and you pay twice the amount right. You do not want that to happen all application which use MQTT, and which use these utility bills and energy metering for instance. You do not want those things to appear twice. So, this is a nice thing you can actually use it with exactly once.

A small transport overhead and protocol exchanges minimize to reduce network traffic, a mechanism to notify interested parties when an abnormal disconnection happens, last will and testament right if you recall that is what it says a mechanism to notify, interested parties when an abnormal disconnection occurs. So, it is a beautiful abstract and speaks the all about the protocol itself, you can go and read this document in great detail cover it in great detail it is does mention about all the messages that we look at. For instance just going through the table of contents, organization of MQTT, terminology, the data representations control packet information, MQTT control packet information, then MQTT control packets, connect message which we saw last time, connack acknowledge connection request publish, publish message pub back publish acknowledgement, pub break publish received right.

Then pubrel publish release, pub comp publish complete, subscribe to topics, sub back subscribe acknowledgement, unsubscribe unsubscribe from topics, un sub back unsubscribe acknowledgemen, ping raq ping request and response ping resp response, disconnect notification, and quality of service 0 1 2 topic names and topic filters and handling errors and authentication of clients by the sever, authorization of clients by the server, authentication of the server by the client; so all of this falling under this category of 5 which is the security right.

I would say you can handle everything, but it is almost you know hard for you to handle this security part right because the thing is protocol we can I mean I can show you many demonstrations, but that is not going to take us I am sure you will pick it up very fast, but you must spend sufficient amount of time on security, I can tell you this. If IoT has to succeed because of you or because of us or anyone of us here the only way by which it can actually become a big success is if you handle security correctly.

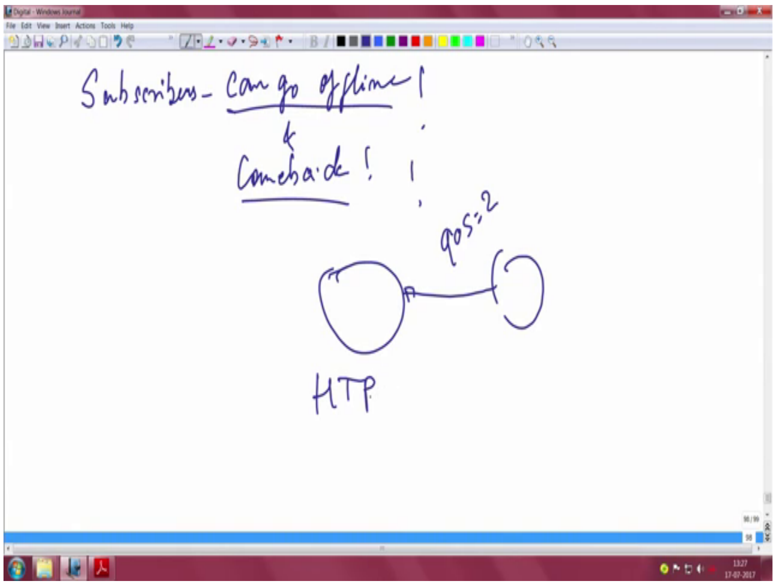
It is big pillar energy is a big pillar, security is another big pillar, these two requirements have to be met. You have to start innovating start looking around how you can get in security into the IoT systems. So, IoT without minus security is not going to take you very far IoT minus handling power is not going to take you far, IoT without analytics and proper data understanding any information is not going to very far. So, you need data analyst to take care of all the data that is being generated.

Data handling has to happen on several gateways and you must be able to handle secure data communication. So, security is a big pillar you need ten year lifetime. So, you need energy management power management also. So, these pillars have to be working together in order to make IoT a great success. So, please spend sufficient significant amount of time understanding this chapter 5. I will assist you in a in making a few small demonstrations with our project staff Miss Tejaswini has spent sufficient time, and we will show you quick demos of these things at the moment right we will finish that, but it does not solve the whole problem you need to understand this thoroughly sit down and understand this chapter thoroughly and of course, we will be happy to take questions on that.

So, this is very very important. So, pay attention to chapter 5, chapter 6 is web socket anything with respect to real timeless bidirectional data transfer with respect to use of MQTT in real time environments. Bidirectional transfer means you have to use you have to understand chapter number 6 which is indeed web socket if time permits we will try and see if you can spend time there. The rest is all straight forward there is confirm and conformance targets and MQTT server MQTT client and so on.

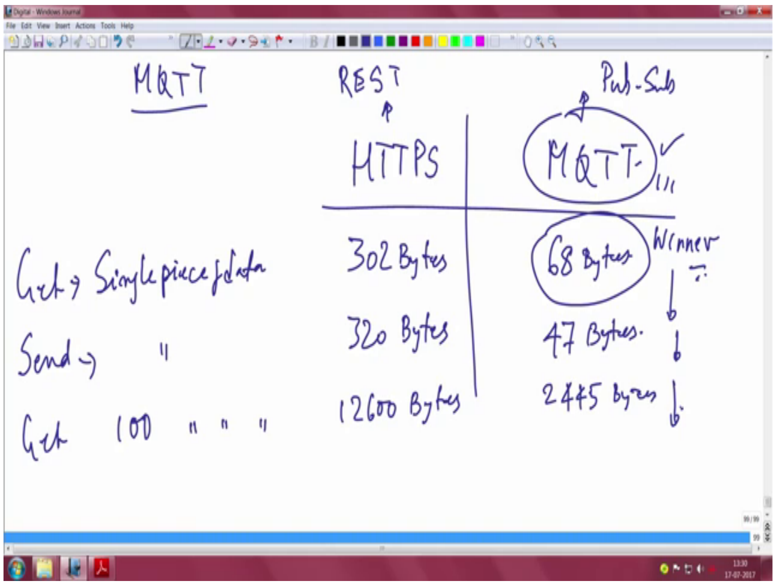
But I think it is important for you while I stress that you need to understand the security chapter in great detail, we have to also I want to draw your attention really to this very good extract which was written about MQTT itself right. It says small code footprint is required and network bandwidth is at a premium you should use this protocol. So, why MQTT is actually listed is actually listed here right and it also says it runs over TCP IP and so on ideal in many situations it says. I want point you to something I saw on the web, and I though you should just you know quickly show you some numbers. If you get I will down make a comparison of http sorry.

(Refer Slide Time: 09:55)



Let us take a new page so that we will be able to understand this thoroughly.

(Refer Slide Time: 10:07)



MQTT	REST ↑ HTTPS	Pub-Sub ↑ MQTT
Get → Single piece of data	302 Bytes	68 Bytes
Send → "	320 Bytes	47 Bytes.
Get 100 " " "	12600 Bytes	2445 bytes

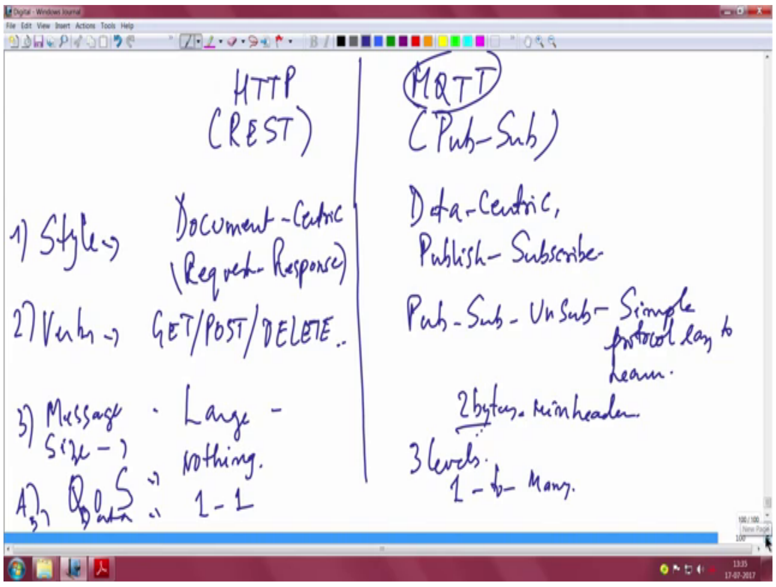
Let us take a new page and let us start with MQTT and it is comparison to let us say I will put http on one side and MQTT on the other you know this right this is pub sub this is it a rest representational state transfer to paradise. You want single piece of data there is a nice comparison you need 302 bytes, this is information I got from the web, but you can actually experiment this and try to find out yourself. You need 68 bytes 68 bytes single if you want to this is to get. Suppose you want to send right if you want to send single piece of data this is 320 bytes and here it is just 47 bytes. If you want to get 100 pieces of data get 100 pieces of data this is 12600 bytes and this is 2445 bytes right.

So, you can go on like this right which is a clear indicator that if MQTT is a clear winner it is a winner and it actually justifies what was written in the abstract particularly, but if you are looking at sensor data sensor networks and sensor data therefore, why MQTT if you ask yourself well one of the reasons why MQTT is because of it is extremely small data packet size, everything is sent in binary form and no comparison at all to https http protocol is particularly architecture. Remember you do not want to compromise on security, I keep telling you this very important you have https in the case of in the in the in the http world you need something very similar in the MQTT world.

Therefore the chapter on MQTT from the standard is a very critical part. So, you have to keep that chapter and therefore, find out how you can add security to that to this particular. So, no compromise no compromise in security, but at the same time reduction in the you know I would say reduction in the payloads, gives you the flexibility of trying to ensure that it is highly optimized for sensor networks, it is optimized for low band with links, it is meant for optimized for low footprint systems particularly in the IoT world.

Which is essentially doing some sensing and some communication some computing communication and some control very specified activity extremely good, because it has that publish subscribe article paradigm and it can do a 1 to many which was not possible. In the case of https because it uses TCP, the complete dealing between the publishers in the subscriber passing through the broker is the great advantage. So, all these things have to be borne in your mind when you try to choose pick MQTT as against other protocols which may be useful right. So, this is one thing.

(Refer Slide Time: 14:17)



	HTTP (REST)	MQTT (Pub-Sub)
1) Style →	Document-Centric (Request-Response)	Data-Centric, Publish-Subscribe
2) Verbs →	GET/POST/DELETE..	Pub-Sub-UnSub - Simple protocol easy to learn.
3) Message Size →	Large - nothing.	2 bytes. min header
4) QoS →		3 levels.
5) Data	1-1	1-to-Many.

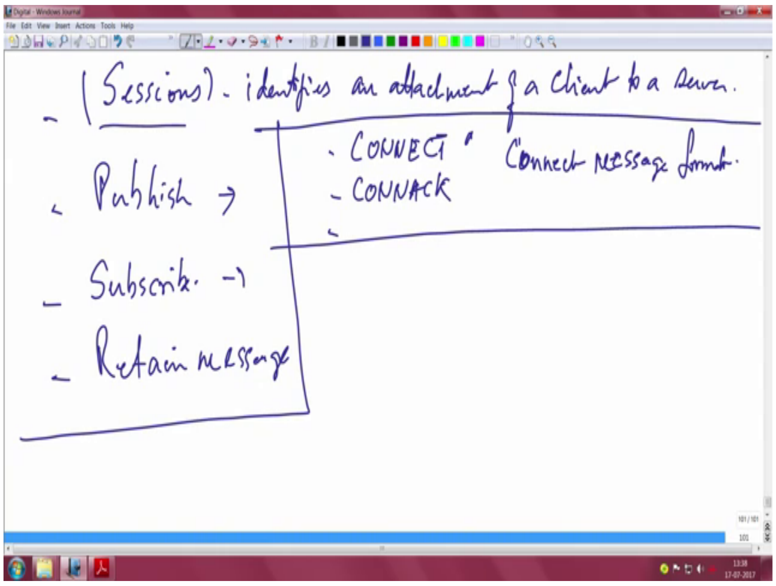
Just continuing on the HTTP restful I would say rest architecture and MQTT the pub sub architecture, just let us quickly finish up this what are the things that you have here in terms of works. So, basically before you go on to that, this is in if you look at this is more like style is more like a document right it is more document lot of things written update formatted well is more I would say more like a nice beautiful document every time a page is uploaded I mean downloaded onto your browser, you would see it make a like a beautiful document it is like a document centric system, and there is a request and then there is a response right in the http case.

But here this is more I am looking for certain sensor data right. So, it is more data centric more about numbers, more about sensing, more about data that is going through in terms of whatever protocol that you use, it is more about the data using data centric and it is indeed using publish subscribe all right this is one. The second thing is if you look at some of the verbs that are used in the m q in the http world, you have get you have post then you have post, then you have delete and so on right the many more such things get post delete, put is also there post put and so on anyway that is all these are all the verbs which are there.

Here you have very simple pub, sub, un sub right and so on very simple. Simple protocol very simple protocol easy to learn right pretty easy to learn, and this is message size if you take if you take message size this is large, large I would say this is just it can be as small as even two bytes. Minimum header means I am not got into the header detail, but if you go through the document you will see the header detail, it can be a small as minimum header can be even as small as this. What about quality of service? Quality of service right you cannot say how you want your document to come alright. So, essentially there is nothing here there is nothing here right, but here you have 3 levels right and if you look at data distribution, data this is mostly one to one here it is one to many right because of the decoupling you are able to do one to many.

So, therefore, there are major advantages if you use. So, I am trying to say why MQTT if you ask your questions, these are some of the super comparisons that you can think of when you can actually talk about MQTT. So, this is in some in some term in some form something very useful for you to keep in the back of your mind before you go. But you may now ask another question which may be of interest to us, but before I go on let me just quickly put down a few things that sort of clear the discussion on MQTT, you must note that session is an important thing right you must look at sessions.

(Refer Slide Time: 18:56)



- (Sessions) - identifies an attachment of a client to a server.
- Publish →
- Subscribe →
- Retain message

- CONNECT ' Connect message format.
- CONNACK

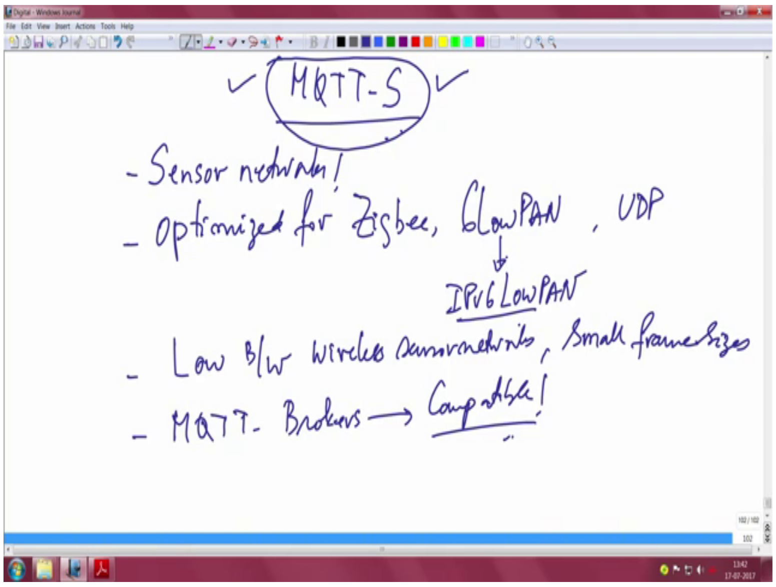
So, I want you to look at sessions identify the attachment basically.

Well, I copied this from the standard document. So, you have nothing to you do not have to worry about anything here. So, just had to look at session attaches and identify the attachment of a client of a client right to a server, and all communication between client and server takes place as part of the session that is very important. Then you may also want to look at publish, I do not want to elaborate this we have done this earlier subscribe, retain message. So, I would say look up these things very important because they are all the ones that will help you understand several things, nice things about the protocol all right.

So, this is one thing and then there are number of messages do not forget to look up MQTTs messages connect, then you have connack we have already seen this, but I want you to look up all these things and then that will allow you to understand the protocol very well. Then there is a format there is a packet format, format you can easily look up I do not want to spend time looking at that the this is a message formats for each one of them connect for instance, connect has a message format. So, there will be a message type then there will be all kind all different fields which are there. So, I want I do not want to spend time there I except you to look up that thing.

But before we move on to anything with respect to security which I promised, I show you some demo though I should also tell you something about how MQTT obviously, all this with respect to MQTT is all fine, if you talk about systems like you know the which have capabilities like.

(Refer Slide Time: 21:19)



✓ MQTT-S ✓

- Sensor networks!
- optimized for Zigbee, 6LowPAN , UDP

$\downarrow$

IPv6 LowPAN

- Low B/W wireless sensor networks, Small frame sizes
- MQTT- Brokers $\longrightarrow$ Compatible!

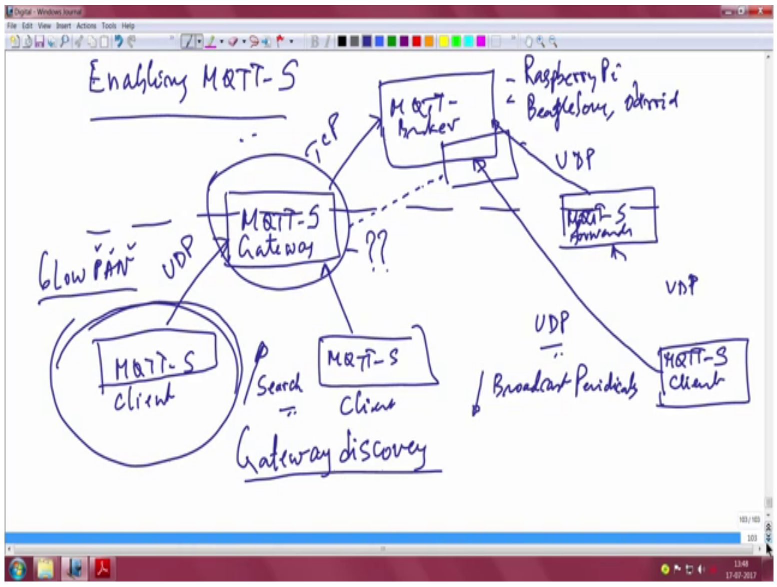
Let us say a raspberry pi right or code raid board or boards which are like beagle bone black arduous boards, beagle board, beagle bone black raspberry pi so on which actually ran into or variants of some Linux essentially. How does MQTT work on very small pieces of hardware how does it work? Because you are talking about sensors which are sensing some environment and they want to push data using MQTT, small little dots right. They have a small vial ultra low power interface it could be zing b, it could be Bluetooth or one of them and it mostly it is zing b let us say because zing b is very popular for building small sensor nodes and it is a very popular.

So, if you are using that, how will you track getting all this stress? You have a huge application stack MQTT stack with all these commands which we discussed and all that; obviously, it would not going to work. So, you have to pay attention to another type of small variant of the same protocol, which is called MQTT s there is a full specification for MQTT s this runs on embedded plat forms. So, if you ask me what is MQTT s, MQTT is meant for sensor networks it is called MQTT s. It is optimized for zigbee well known zigbee it uses 6 low pan I am sure you know already what this means, 6 low pan is IP v 6, for IP v 6 for low power personal area networks ok.

This is IPV 6 adaptation for constrained platforms 6 low pan and you do not have to use TCP you can use UDP. Traditionally MQTT uses TCP and you can you actually use it on UDP. It is meant for low bandwidth, it is meant for low bandwidth wireless sensor networks right and frame sizes are small I will say small frame sizes small frame sizes and simple devices right and it is what is good about is it is really compatible with the normal MQTT brokers compatible that is the beauty right; obviously, if you want to make it compatible, it is not going to be straight forward for. You have to build a nice system which I want to draw and show you the picture related to how you can actually enable enabling I will say enabling MQTTs.

How will you do this, essentially the way it goes let me just quickly draw a picture for you which will allow us to understand it better. You have MQTT s client is let me put it inside this right. So, that is right way to do essentially we will talk to know what is known as a MQTT s gate way, this gateway in turn will now talk to MQTT broker ok.

(Refer Slide Time: 25:02)



That is the nice thing. In fact, this gateway can actually be part of the system. So, I will put a dot here showing that this system while it is nicely demonstrated shown this way, you can actually push this gateway inside this hardware itself which is it could be I would say when I talk about an MQTT kind of a broker hardware, I am referring to systems like a raspberry pi please note raspberry pi or I may be referring to beagle bone right beagle bone or I could be referring to a droid on droid system and so on which having I mean in terms of they are low power still, but they have a better functionality compared to extremely small devices like this. So, you can have these MQTT clients essentially this is a client, this is also a client and they are all connecting to this nice gateway and see the beauty this now run UDP, they can run UDP and up to this layer they actually run 6 low pan right.

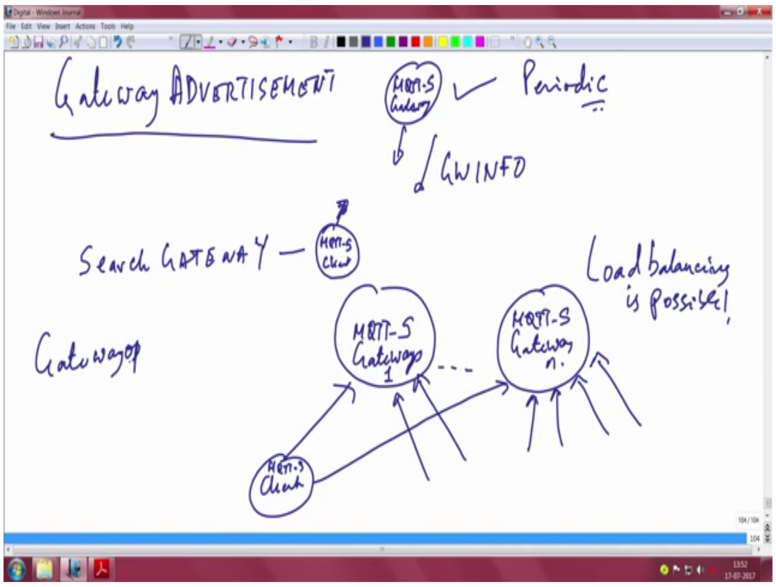
IPV 6 for low power personal area networks, there is an RF c on this very popular implementation of IP v 6 for consistence environment, is not it 6 low pan it works very well on zigbee. So, that is been zigbee MQTT. This can be TCP right this can already TCP, there are other ways by which you can do that you can also have if you do not want to pass it through a gateway you can also use what is known as MQTT s forwarder.

So, there is another possibility you can also use MQTT s forwarder and so clients, what kind of clients? MQTT s clients can connect on UDP and connect to this forwarder, forwarder will not change anything it simply change it will simply forward that packet on UDP to the broker and the broker in turn will change it and maintain a certain information it is just a simple forwarding system. So, it has to be UDP all through and it in tern converts into a client connection and so on.

So, either you can essentially passes through a forward in which case you may now ask a question, why do I need this forwarder at all I can as well send directly to the broker surely you can also send it directly to the broker over UDP. So, there are many many scenarios that you can try, that is the point I am trying to say is that there are many scenarios in which you can use it, nobody stops you from trying out your own scenario. If you are able to configure and cleverly write some scripts why not you can as well do this thing nicely ok.

So, there are brokers there is a gateway and all that, now it is not that this is just going to work like this, what is special about MQTT s, why is it written for this means you have one difficulty already here right, this MQTT s protocol which is running on this very small tiny systems have actually got to find this gateway; where is it they have to do which means gateway discovery is important. Either the gateway can broadcast broad cast periodically or clients have to search right.

(Refer Slide Time: 30:47)



Either the clients have to search or the gateway has to broad cast periodically correct. So, which brings us to very simple things again gateway advertisement exists a d v e r t i s e ment gateway advertisements exists ok.

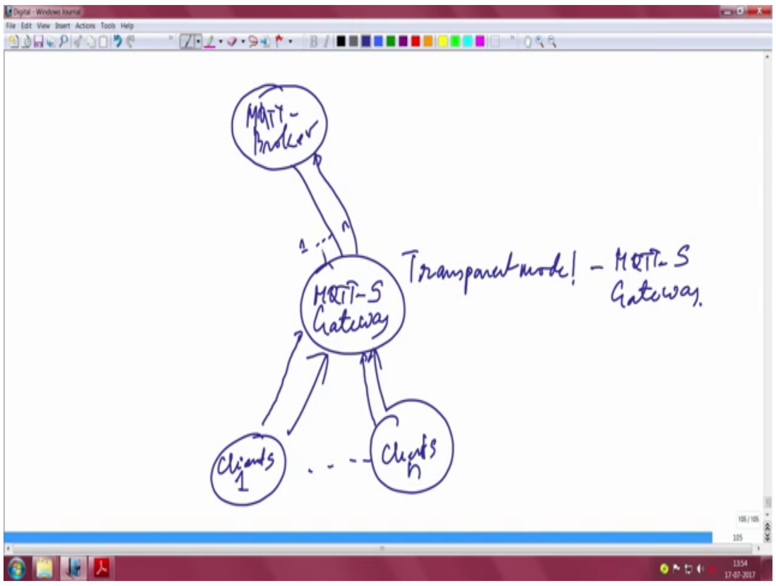
So, this is coming from the gateway gate way is actually giving these advertisement this is the MQTT s I will say very important MQTT s gateway and if you are the client is trying, is going to do a search for gateway for some reason you have to put everything in cap, and you get. If you do a search that is this guy who MQTT s client, let me not muddle it all right it will little bigger.

In fact, I should not write bigger than the gateway circle, because gateway is expected to be little more powerful this is MQTT s client. Client is sending it out for in response to search gateway, this gateway can actually issue back what is known as GW info. So, now, this completes the story you can have periodic transmission of gateway advertisements. If somehow the MQTT s client miss that periodic advertisement, it can also issue a search gateway for which the gateway can respond with GW info, that is what essentially this whole thing means.

There is something nice about these gateways; see is another problem these MQTT s gateways will encounter will be that they will be connected to they will have connections data coming from many many sensor nodes, because now you are talking about several hundreds and hundreds of sensors which are trying to publish the data right which means clients what kind clients MQTT s clients can actually connect to multiple gateways so that load balancing of when I say load balancing, basically means data load balancing opps it is not coming out well let me rewrite balance balancing is possible.

So, now you can see it is it actually skills quite well. So, that is another thing. So, this is one other important thing there is another final thing that you may want to actually note that this gateway itself operates. So, sorry this gateway itself operates in two modes right operates gateway operates, I will clean up this I will go to the next sheet.

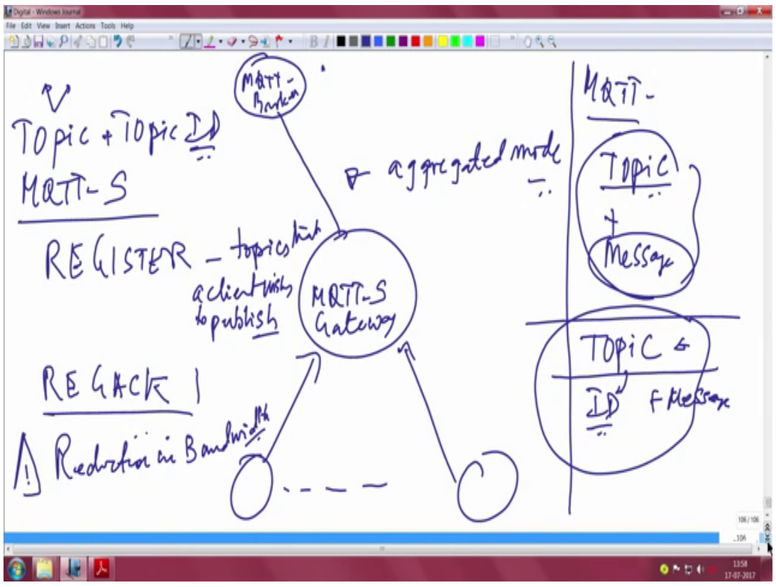
(Refer Slide Time: 34:48)



And then we will re write it MQTT s gateway, and this is connecting to what? It is connecting to the MQTT broker not the s broker, but the normal broker right it is connecting to the normal broker and this MQTT s is getting several connections from MQTT s clients perfect clients, this would be client 1 client 2 client and so on and so forth.

So, this is one let me put another arrow here, show another one here and all that now you can do it two ways, you basically create a connection between the server and the broker for every client that connects. If there is one client connecting there will be one connection if there are n clients connecting there will be a n connections from the MQTT s to the broker. Clearly this is called transparent mode this is transparent mode. Another mode is there which is called the aggregated mode this is mode one of MQTT s gateway another mode is there and that mode corresponds to what is known as the aggregated mode where you maintain. So, the picture is quite simple. So, what you do is I should remove this now somehow I do not feel like removing this. So, I will take another sheet simple.

(Refer Slide Time: 36:48)



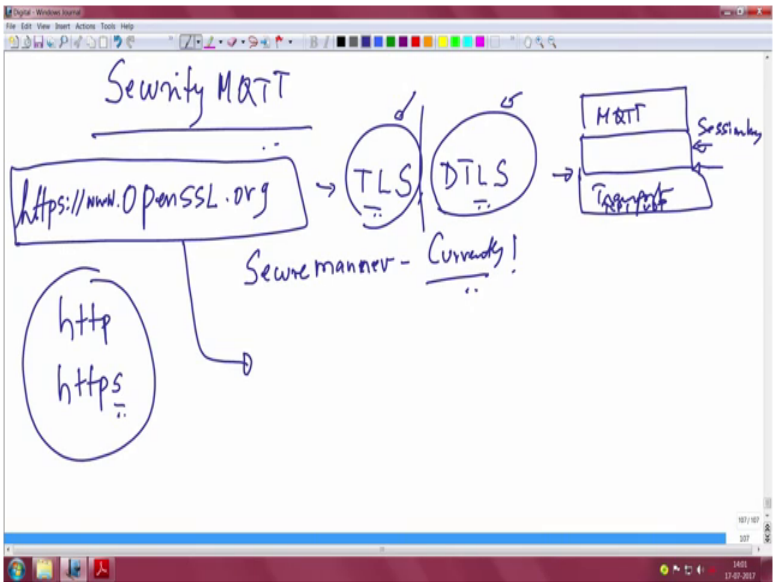
I will take MQTT s gateway and multiple clients connecting, you can have a single connection to the MQTT broker this is called aggregated mode that is the key thing. There is one last point which when I was talking to Tejaswini it actually occurred, that we should we can do several interesting thing. In fact, MQTT s actually supports that what it says is look carefully at your original MQTT protocol, original MQTT protocol says for every topic, topic plus message is given right you are doing topic plus message in MQTT why do you want to send keep sending topic every time in the MQTTS world.

What you do, you associate you send the topic only once with an ID toward and then you only start using the ID do not use anymore topic do not say topic message topic do not keep using the good topic anymore just use the topic ID and keep publishing the data or subscribing to the data which is the clear indicator and again a clear the situation where you optimize for sensor networks and constrained environments that is the key take away of the MQTT s protocol.

In order to facilitate this topic ID topic plus topic ID one time and keep reusing this topics there is a requirement actually topics there is there are two things are there is a long topic and then there is a short topic, you can use both. Actually ID s can be either long ID or short ID. In order to facilitate the possibility of using ID s and then the message topic ID plus message instead of topic plus message, the MQTT s supports an additional set of messages which is called register this is to register the topics that a client requires to publish this is to register the topics that a client that a client wishes to topics that a client wishes to publish wishes to publish and exactly if you send out a register message what you will get back is a regak, which essentially allows you to complete you and again essentially facilitating reduction in key, what is the key here? Reduction in bandwidth.

Now, go back and look up at the original abstract given by MQTT oasis standard, it covers everything that we discussed in it is completeness right.

(Refer Slide Time: 40:52)

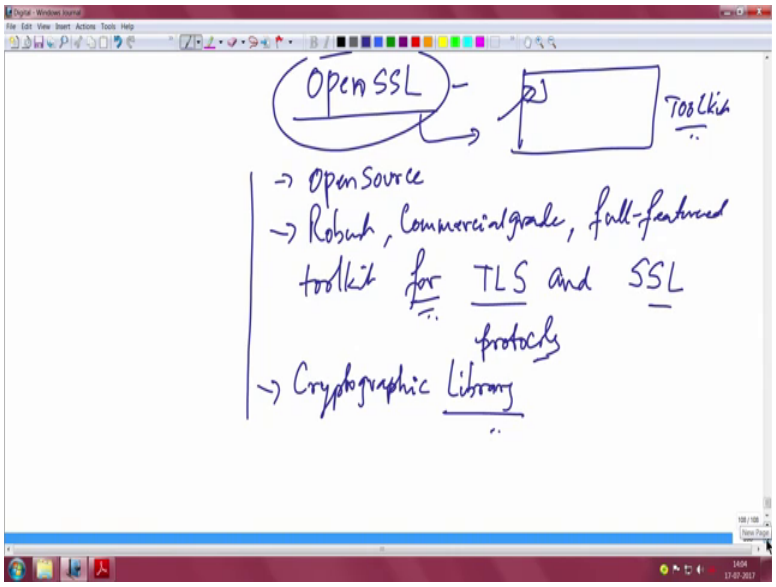


So, this is the key point of a MQTT. Now finally, let us now let us move on to understand the security aspects of MQTT security of security in MQTT I am not going to give you a full story here, but indeed we will do demonstrations of security in MQTT and you understand everything with respect to that by seeing these demonstrations, but before we switch to demonstrations, I want you to always before you get into anything definitely go to OpenSSL; OpenSSL dot org you must visit https cool colon www OpenSSL dot org.

I think this is going to hold the key to transport layer security, either TLS or datagram transport layer security. These are the two definitely protocols which will allow you to communicate between an MQTT client and the broker in the most secure manner secured manner currently, the most secure manner you must look at that what does essentially it means is if you take an application layer where MQTT is running you have the session layer which essentially requires a session key a session key and then you have the transport layer essentially like TCP or UDP. You are essentially using TCP and UDP in a secured manner, which essentially will create the transport layer security and either if you using TCP TLS and if you are using UDP to be DTLS.

Essentially it is sitting here in these a little a little above I would say somewhere exactly at the transport layer, and completely giving you secure communication. And how do you know whether you are doing secure communication well you are not going to use http anymore you going to use https from now onwards, clearly indicating the TLS is playing a big role in establishing the http connection. It does not mean UDP does not have that is why I wrote here DTLS is also a very important thing they go hand in hand, but what is OpenSSL what is it going to help you, and why should you look at this because anything with respect to security, will essentially a comp you have to just note this one picture.

(Refer Slide Time: 44:08)


OpenSSL – Toolkit

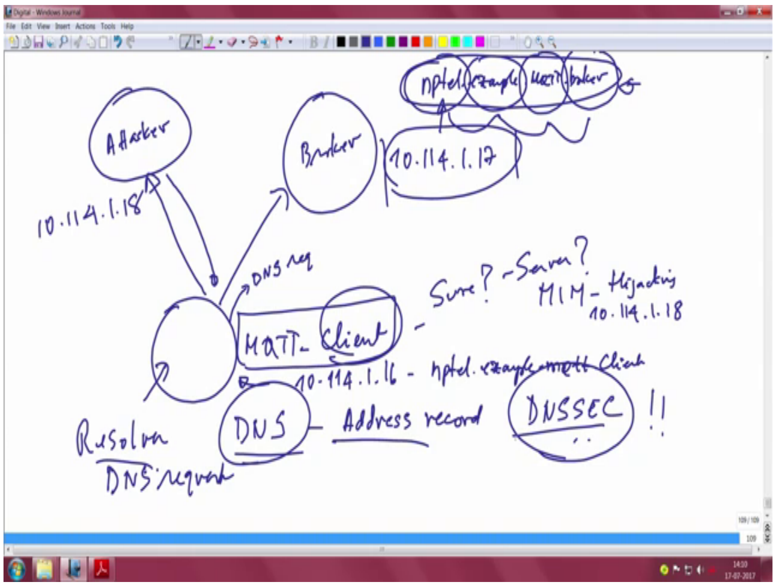
- Open Source
- Robust, Commercial grade, full-featured toolkit for TLS and SSL protocols
- Cryptographic Library

So, I will write OpenSSL and I will put a picture which I expect that you will remember throughout with respect to understanding OpenSSL in.

So, what is OpenSSL? OpenSSL is nothing like a box, which has a tool kit remember OpenSSL is like a box which has toolkit what all do you have screwdrivers cutting pliers right nose pliers and so on and so forth. These actually make a mechanics toolkit quite like that OpenSSL is a crypto cryptographic toolkit it is a tool kit. So, this tool kit will allow you to do several things, it is not a protocol please note OpenSSL is not a protocol a protocol use OpenSSL, but OpenSSL itself is not a protocol because OpenSSL is purely a toolkit what is good about OpenSSL? iIt is open source, that is why it is called OpenSSL. It provides robust commercial grade right commercial grade full featured toolkit as I mentioned it is a tool kit for this is important it provides a tool kit for TLS transport layer security and secure sockets layer SSL protocols these are protocols that is the key point I am trying say.

So, it is also a general purpose cryptographic library library and you will see a lot about it. In fact, it is one of the most active websites which has lot of updates updates coming up here. Why is OpenSSL why am I making noise about OpenSSL? I want to draw your attention to a picture which I am going to draw in the MQTT world, which will allow you which will give you an understanding of why this is important take an MQTT client ok.

(Refer Slide Time: 46:58)



MQTT client which wants to connect to in MQTT broker this is broker. What will you do? You will specify the IP address of the broker here right before you do a publish. So, you will say let us take an example ten dot 1 1 4 1 dot 16 is the let us say the IP address of this client and ten dot 1 1 4 1 dot 17 has a example for a example is the IP address of the broker right.

Now, the question is this right let us say in this case map to www let us say MQTT no I will just give it a name example, no I will give it to a better name nptel dot example, dot

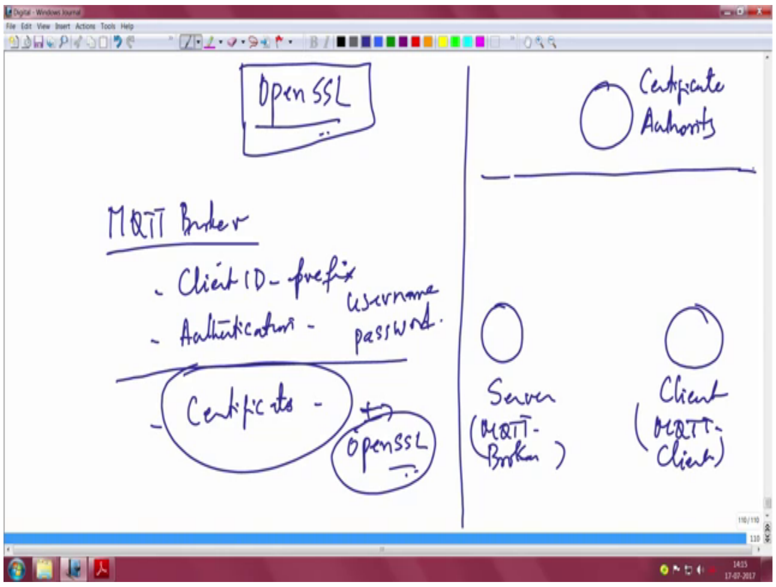
MQTT right something let us say this is the name of the MQTT broker let us put it MQTT dot broker, this is what it maps to let us say. Now what you are going to do and let us say this is let us give it also a name nptel dot example dot MQTT dot client let us give it two names very good. Now first thing that will happen is when the client wants to connect to the broker he might even give this name, nptel dot example. In this case what will happen? DNS will kick in the pitch in and resolve the IP name address the full name to this particular address and there is an address record associated with this address. So, DNS address record will be fetched and it will be given to the MQTT client and the client will make a connection based on, and it will get this IP address because address record will actually give you the mapping of name to address that address will come.

Now, I want to ask you a question, how are you sure how is the client sure how is the client ensure that it is actually connecting to the server, how does it not connecting to the server. In other words there is a possibility that there is a man in the middle which is hijacking and providing an IP address, 10.114.1 dot 18 right and the client is now connecting to the attacker; what is actually happening? when the DNS goes out this attacker listens and quickly responds first attacker is very very aggressive, when the DNS request was goes out for fetching the resolver this is nothing, but the resolver when the resolver issues request DNS request, the attacker comes in even before the broker can respond and gives it is IP and the MQTT client things that he can easily he thinks that this is indeed the broker and connects to the attacker.

Now thinking that it is the broker and starts publishing to the attacker instead of the broker, will people know this and people have found that somehow we should avoid doing this right. So, what people have done is why should we use DNS at all use DNS sec secure DNS this will solve the problem perhaps now let us see how does DNS sec work? First is DNS sec works because the client is not going to look at certificate coming from the route, down to MQTT, down to example and down to the actual server. So, it is going to cross check each time to find out whether this certificates actually match at each and every level starting from the root a top level from the root and an all the levels in the DNS actually it matches everything whether that is matching or not.

That part it will do, but you can actually attacker can be responding too in the same way right the attacker are actually be providing u certificate after certificate which essentially there is no way of catching whether you are connecting to the actual broker or not by just using DNS sec that is not going to solve your problem therefore, people have sort of abandon DNS sec and decided that everything they want to do is with respect to open SSL.

(Refer Slide Time: 53:14)



And therefore, security in the OpenSSL becomes a very important kit that you should understand and pay a lot of attention to MQTT s security related settings. I will now shift to the demonstration mode on my laptop, and we will see quickly several demonstrations on how we are able to handle set security related things I give an overview. And I will show you quickly several things there are basically 3 ways by which you can secure your MQTT broker, all settings with respect to whatever we are discussing now key is with respect to MQTT broker ok.

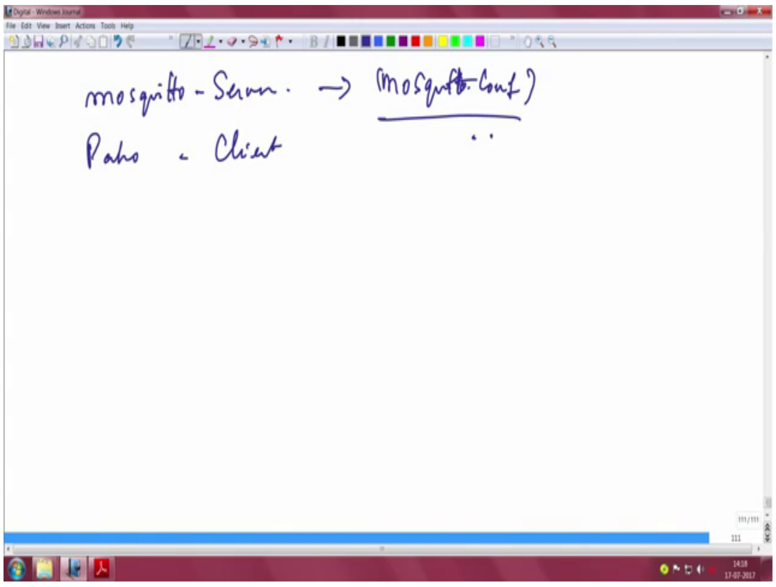
First thing that you can do is you can do client ID prefix you can do a prefix this is a very important step that you can do. The second thing is simple authentication which means please put the username and password right put the username and password username and password ifs. Third thing is use huge certificates, digital certificates both for authentication as well as for creating sessions and ensuring that data is encrypted and send between the client and the broker.

Now, if you look at certificates keep this picture in mind before demonstrations start remember 3 entities in the certificate world you and using openssl. So, when you say certificates you are talking about the use of toolkit OpenSSL, 3 entities have to be made a note. One is called certificate authority the second one is the server which server do you mean MQTT server, why I am using the word server I better use a better e name that we know very well MQTT broker right and client. When I say client, I mean MQTT client. This OpenSSL commands in command line will allow you to create all these these certificates for all these 3 entities.

You want this certificate authority because it is independently going to inform the client whether the server is connecting is indeed the right server or not. Remember DNS sec in DNS sec also you did not know all though you verified certificate after certificate from the top level from the root or downwards, upwards event you were able to look at the certificates it did not prevent that the certificates where while they were long live alright it may have actually come from the attacker. Here there is an independent certificate authority, who will verify everything for you and actually let the client connect to the server and the client can satisfy itself that it is connecting to the right server.

Well there is a possibility that the server may also want to check, whether it is connecting to the right client that is optional though, but you have to note that there is independent. So, that is still optional. So, these are essential elements you have a certificate authority, and then this is a server and the client. So, let us now see how all of this is actually done quickly, and we can also look at some of the, to actually demonstrate whether this whole system using certificates actually work very well all right. So, point really I want before we go on to see the demonstration on the laptop, you have to note that all we are using mosquito.

(Refer Slide Time: 58:09)



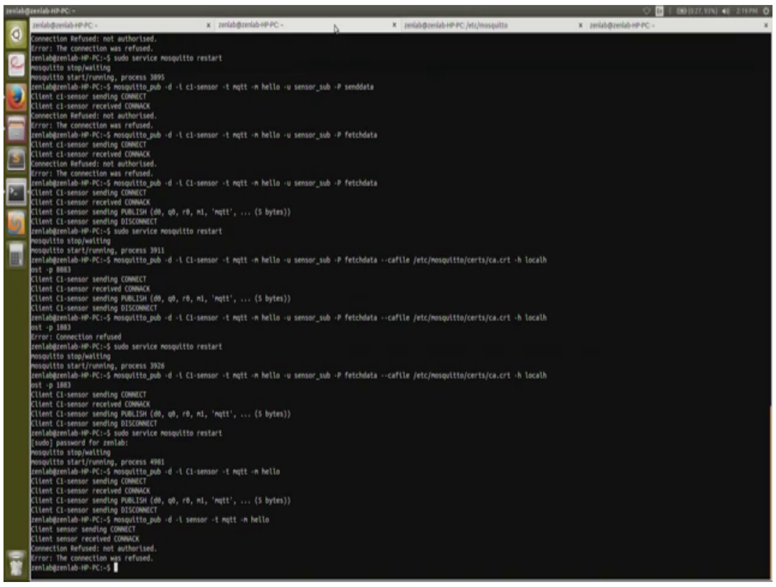
mosquitto - Server. → (Mosquitto.Conf)

Paho - Client

So, mosquitto. So, let me write it well mosquitto server which is nothing, but the broker and paho is the client all modifications are being done to mosquitto dot conf. Please note this file name and this is all part, you are going to do the sequence of demonstrations will start with client ID followed by password, then followed by certificate based security configuration for brokers.

So, let us start with the first one, the first one is let us start to the other screen the screen here you can see is starting with let us go slowly.

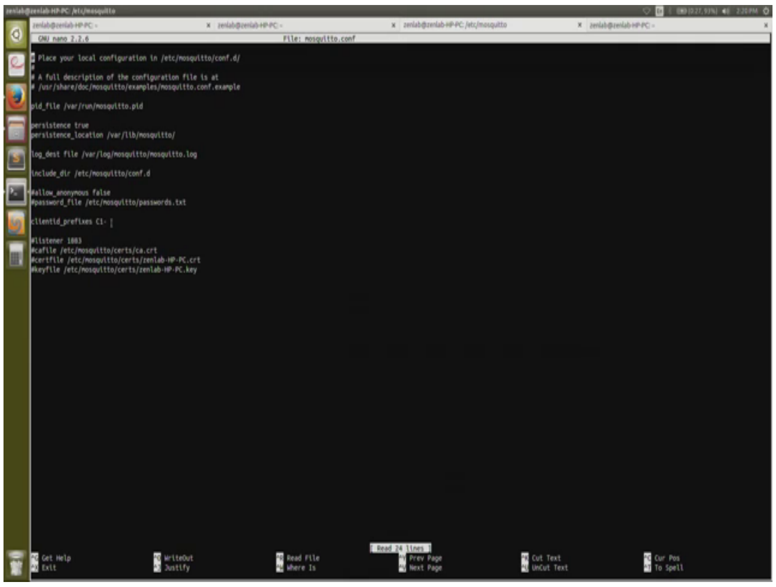
(Refer Slide Time: 59:09)



```
Connection Refused: not authorised.
Error: The connection was refused.
zenlab@zenlab-HP-PC:~$ sudo service mosquitto restart
mosquitto stop/waiting
mosquitto start/running, process 3895
zenlab@zenlab-HP-PC:~$ mosquitto_pub -d -i c1-sensor -t mqtt -m hello -u sensor_sub -P senddata
Client c1-sensor sending CONNECT
Client c1-sensor received CONNACK
Connection Refused: not authorised.
Error: The connection was refused.
zenlab@zenlab-HP-PC:~$ mosquitto_pub -d -i c1-sensor -t mqtt -m hello -u sensor_sub -P fetchdata
Client c1-sensor sending CONNECT
Client c1-sensor received CONNACK
Connection Refused: not authorised.
Error: The connection was refused.
zenlab@zenlab-HP-PC:~$ mosquitto_pub -d -i C1-sensor -t mqtt -m hello -u sensor_sub -P fetchdata
Client C1-sensor sending CONNECT
Client C1-sensor received CONNACK
Client C1-sensor sending PUBLISH (d0, q0, r0, m1, 'mqtt', ... (5 bytes))
Client C1-sensor sending DISCONNECT
zenlab@zenlab-HP-PC:~$ sudo service mosquitto restart
mosquitto stop/waiting
mosquitto start/running, process 3911
zenlab@zenlab-HP-PC:~$ mosquitto_pub -d -i C1-sensor -t mqtt -m hello -u sensor_sub -P fetchdata --cafile /etc/mosquitto/certs/ca.crt -h localh
ost -p 8883
Client C1-sensor sending CONNECT
Client C1-sensor received CONNACK
Client C1-sensor sending PUBLISH (d0, q0, r0, m1, 'mqtt', ... (5 bytes))
Client C1-sensor sending DISCONNECT
zenlab@zenlab-HP-PC:~$ mosquitto_pub -d -i C1-sensor -t mqtt -m hello -u sensor_sub -P fetchdata --cafile /etc/mosquitto/certs/ca.crt -h localh
ost -p 1883
Error: Connection refused
zenlab@zenlab-HP-PC:~$ sudo service mosquitto restart
mosquitto stop/waiting
mosquitto start/running, process 3926
zenlab@zenlab-HP-PC:~$ mosquitto_pub -d -i C1-sensor -t mqtt -m hello -u sensor_sub -P fetchdata --cafile /etc/mosquitto/certs/ca.crt -h localh
ost -p 1883
Client C1-sensor sending CONNECT
Client C1-sensor received CONNACK
Client C1-sensor sending PUBLISH (d0, q0, r0, m1, 'mqtt', ... (5 bytes))
Client C1-sensor sending DISCONNECT
zenlab@zenlab-HP-PC:~$ sudo service mosquitto restart
[sudo] password for zenlab:
mosquitto stop/waiting
mosquitto start/running, process 4981
zenlab@zenlab-HP-PC:~$ mosquitto_pub -d -i C1-sensor -t mqtt -m hello
Client C1-sensor sending CONNECT
Client C1-sensor received CONNACK
Client C1-sensor sending PUBLISH (d0, q0, r0, m1, 'mqtt', ... (5 bytes))
Client C1-sensor sending DISCONNECT
zenlab@zenlab-HP-PC:~$ mosquitto_pub -d -i sensor -t mqtt -m hello
Client sensor sending CONNECT
Client sensor received CONNACK
Connection Refused: not authorised.
Error: The connection was refused.
zenlab@zenlab-HP-PC:~$ 
```

Now, you basically want to you basically what we are done here is you are publishing with a ID, which is called C 1 dash sensor the topic is MQTT the message is hello. So, let us see what happens to this, you can see that this was done successfully. Now what we will do is remove the client prefix and transmit it again and see what happens. So, you can see if you. So, very simple thing you can do, configure in mosquito dot conf the client prefix. So, let us open that file and see what exactly happens in that file.

(Refer Slide Time: 59:58)



```
  GNU nano 2.2.6                         File: mosquitto.conf

# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /var/run/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

#allow_anonymous false
#password_file /etc/mosquitto/passwords.txt

clientid_prefixes C1-

#listener 1883
#cafile /etc/mosquitto/certs/ca.crt
#certfile /etc/mosquitto/certs/zenlab-HP-PC.crt
#keyfile /etc/mosquitto/certs/zenlab-HP-PC.key

                                  [ Read 24 lines ]
^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

So, let us see you can see here it is nicely point in let us take the cursor, there take the cursor there, cursor there, exactly t there come to the cursor here yes you can see there the cursor is pointing to client prefix C 1; that means, it will have to look for C 1. So, it is like security through obscurity right this is something like that, so this is the simplest do something but keep it secure this is the first part of the demo. The second thing is with is with respect to passwords. Now in order to get to passwords you have to do allow anonymous falls tell you if it is true earlier it was true now we have made it falls, and then you have to create a password file and in our case we have called it password dot text and there you should mention the password.

So, let us see what actually happens. There again note that we are modifying mosquitto dot conf all right. So, now, I do this we continue with the same exercise you see mosquito pub minus d the prefix is c1, topic is MQTT message is hello then you have the user name is sensor underscore sub and the password is fetch data fantastic it works alright. So, now, let us see what happens we will remove. So, what we will do we will remove and put a wrong password here right. So, you can see that it has refused the connection, has it no it has not refused connection what is the connection what is the password here the right password is this. So, let us see no I think this is some let us now. So, we restart that was an issue. So, let us restart sorry. So, every time you make changes to the file mosquitto dot conf what is important is to stop the demon and restart the demon and you can see now the affect has taken place.

So, this is another lesson that you learn that every time you modify the configuration file you must restart the demon. So, that changes are affected just by making modifications to the conference is not going to take you very far please note this these are very simple things, but they start playing a very important role. Now what you do is go back to this conf file and look at all these 3 settings; you have the certificates essentially the first one is the c a certificate that you see here, the second one is the server certificate and the third one is the server key right these 3 things are there here, the listener port is one double eight 3 and now let us see how this works.

So, now that these certificates have been created you will have to restart the demon. So, we will restart otherwise this affect wont takes place, now we will start putting the right password the right username and the c a certificate path right. So, let us put the path to the certificate yes it is mosquito search c a dot c r t and port is mentioned as 181 double 83 back fantastic it works right. This is the best this is using TLS and indeed it has full security enabled on the this. So, they essentially this essentially tells you that it is fully secured, you have client ID prefix you have passwords, and you also have all the certificates installed so that only a only TLS is now working in is working completely.

Now, you just to complete the story let us remove the c a certificate or give it a wrong path or try something, which is just to demonstrate that without this you can see that it does not connect. So, you can see that in summary these steps have to be done, and you have to ensure that the highest possible level of security actually is enabled for the MQTT protocol.

Now in summary you may now ask a question, how am I going to pull off all this on very constrained devices. One option that you can think of is at least you must be able most microcontrollers today, provide you encryption on the fly essentially they will allow you to encrypt they have the a e s encryption, symmetric key encryption like a i a s in hardware implemented in hardware. So, you should perhaps you know put key during the factory dispatch itself and then have another way of changing that key using bootstrap in techniques that is possible. And these are very very major ways of doing it.

So, please do lookup bootstrapping techniques for change in the default a e s key in order to ensure that on constrained devices like the MQTT a clients minimum you should have is a data encryption in running ok. And there maybe another possibility that you may actually hold the sessions if you are working with if you are transmitting data you know all this essentially means TLS essentially is a overhead in terms of establishing a connection after you.

So, before you actually do any data transmission there is this basic handshake; TLS a huge handshake protocol which happens by you know this certificate exchanges verification by the client the certificate authority are coming into picture and then saying yes everything is fine you can connect and all that, means there is a handshake associated.

That handshake part if you want to avoid and that is a lot of data being transmitted between the broker and the client each time you want to run TLS. Therefore, there are options in MQTT which will allow you to hold you do once. You do the handshake once you keep the session key permanently with you and then use that the session key for all subsequent transmissions.

So, do explore the chapter on security in MQTT specification to understand completely the capability of this exciting IoT protocol.