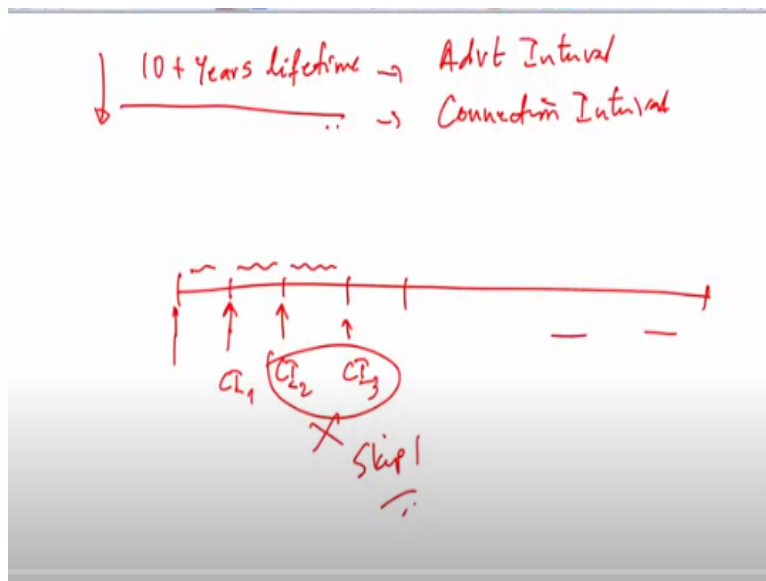


Design for Internet of Things
Prof. T V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science-Bengaluru

Lecture - 42
Bluetooth low energy (BLE) – 02

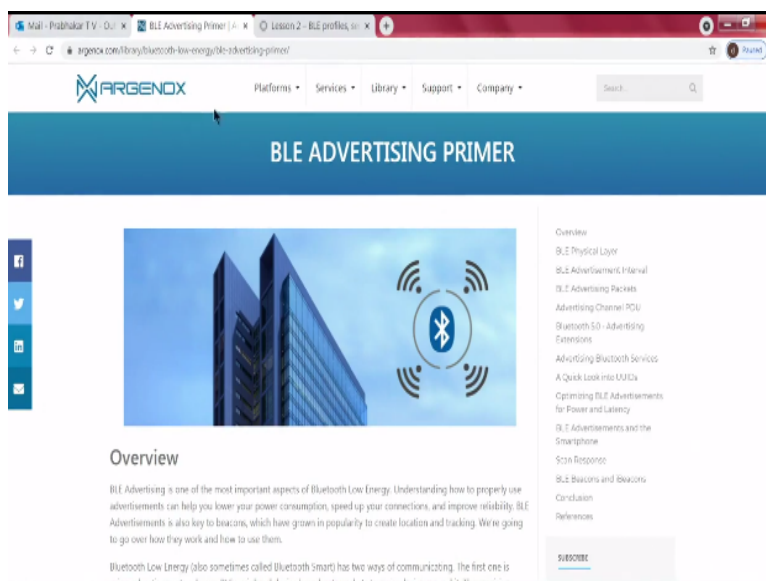
So let us go into some detailing there.

(Refer Slide Time: 00:31)



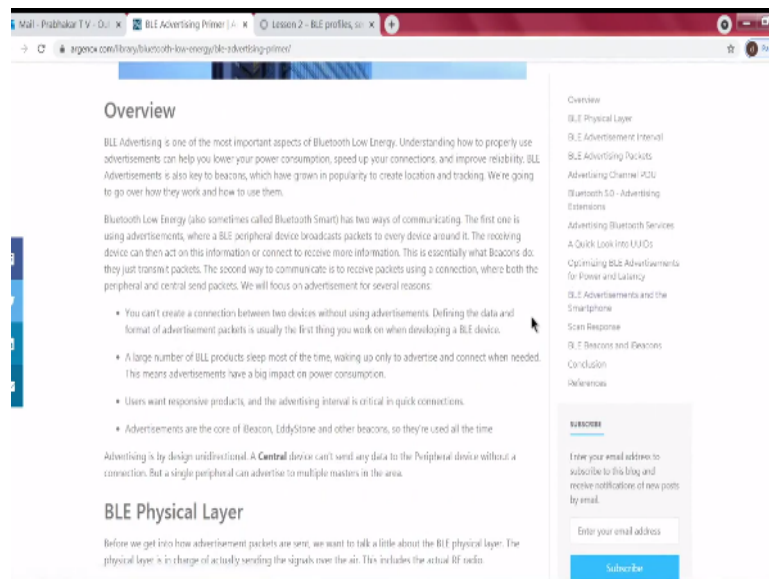
For that, what I thought is best is to read up a document, which I found was extremely useful.

(Refer Slide Time: 00:35)



This is from this company here ARGENOX. And I found that this BLE advertising primer, given by this company is really good. It covers several aspects of it, okay.

(Refer Slide Time: 00:50)



Here is the first part of talking about how BLE advertising is a very important and promising aspect of BLE, and all that. And why it is important to ensure that you configure it properly so that you get low power consumption yet at the same time, you will be able to speed up a connection, if you want to do connection oriented transfer of data.

And also how you can you know sort of improve reliability of these connections and so on, right? That is one part. It also talks to you about the fact how advertisements actually become beacons okay, in a way they become beacons. What is this transformation of an advertisement into a beacon may be intriguing to you, but I will explain that very soon, right? So this is another important thing.

So let us keep this in mind how we can make them into beacons. Then you can read this document, it tells you that it is also called Bluetooth Smart and so on. And it tells you about one important thing that beacons are essentially just transmitters. They cannot, if they are in the transmit mode, they cannot receive anything. And they are configured only for transmit.

In fact, they are kind of systems which are in pure sense only have to, I mean simplex, okay. They are simplex by nature. There is no way by which you can actually get them to do even half duplex. So that is essentially what it means. It just transmit packets, alright. And there can be typically those peripheral systems. And they simply keep sending out beacon packets at regular intervals.

Now the advertisement itself, you need there are two types, right. So one is you do an advertisement, because you want to do a connection, you want to do a connection. And you can also do an advertisement because you do not want to connect. But essentially, they become beacons. And one such, you know industry standard is the iBeacon, apple iBeacon, EddyStone, and so on.

These are all beacon types. These are actually advertisement packets, but they are as I said, they are actually simplex by nature. And therefore, there is no way by which you can actually do any connection. And that is typical of what you want to do in a department will store. The footwear guy will configure it as an iBeacon or as an EddyStone and say that I am only pushing the ad.

I do not want you people to connect back to me, right? You just configured it and it has to keep beaconing that at some interval. And that is what is the advertisement interval, right? So you can do very clever things, by the way. You can look at, you can do human detection and trigger this. You put one camera, and see if you are seeing humans enter the mall, or enter the shop, then you start triggering.

Because radio transmission is expensive. So you may want to do multimodal sensing, and then you may want to start triggering these advertisements, okay. That is also possible. But also you can take it to spiraling levels folks. You can go into let us say the footwear shop, because you saw that beautiful ad in the corridor. You said okay 40% discount on my on a particular type of footwear.

So let me go and explore it. You go inside the shop, it is a big shop. And somewhere, you know you have identified it with your let us say naked eye, the type of footwear

that you are looking for. So you went close to that. When you went close to that it told you something more on your phone. What you did not see in the corridor, you found when it was when you went very close to the footwear.

In other words, you can control the range of this packet range, transmission range by going very close and getting more information about it, alright? Why do you want to do that because if you put it at a extremely low transmission power, the battery replacement can be extended right, because you are not expending too much of battery power.

So all these things will have to be thought of when you are trying to look at designing BLE based embedded nodes, IoT nodes, both for the purpose of just advertisement, advertisement leading to a connection or advertisement in the simplex form, which is essentially for the purpose of beaconing where there are two competing technologies, Apple iBeacons and EddyStone technologies, right?

Both of them are competing. And essentially, you can use that, any one of them.

(Refer Slide Time: 05:44)

The slide compares BLE and Classic Bluetooth. It notes that BLE uses a completely different modulation than GFSK. Classic Bluetooth has 79 channels compared to BLE's 40 channels, and they are spaced differently, making them incompatible. Dual Mode Radios support both by switching modulation parameters and channels.

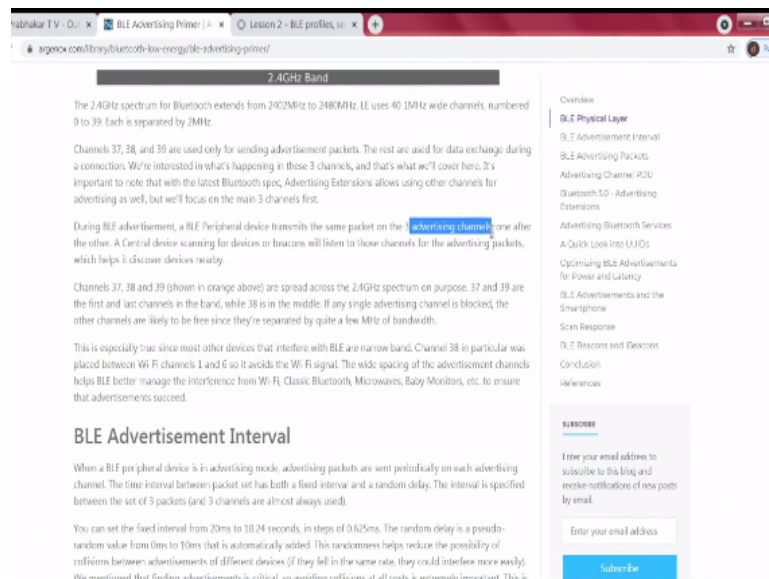
	BLE	Classic	
	BLE	BR	EDR
Modulation	GFSK 0.45 to 0.35, 0.5 (Stable Modulation)	GFSK 0.28 to 0.35	DQPSK / 8DPSK
Data Rate	1Mbps, 2Mbps (Bluetooth 5.0)	1Mbps	2Mbps, 3Mbps
Channels	40	79	79
Spacing	2MHz	1MHz	1MHz

The diagram shows the frequency spectrum from 2402MHz to 2480MHz. It highlights 'Advertisement Channels' (yellow) and 'WiFi Channels' (red) within the BLE band.

So this is about BLE five layer, I would not go into the detail. You can read this carefully, you will know what are the number of channels in the BLE as compared to the classic and all that. This is all well laid out. So I think if you read it once, you will

understand them well. All it boils down to is these are the advertisement channels 37, 38 and 39. I mentioned this to you.

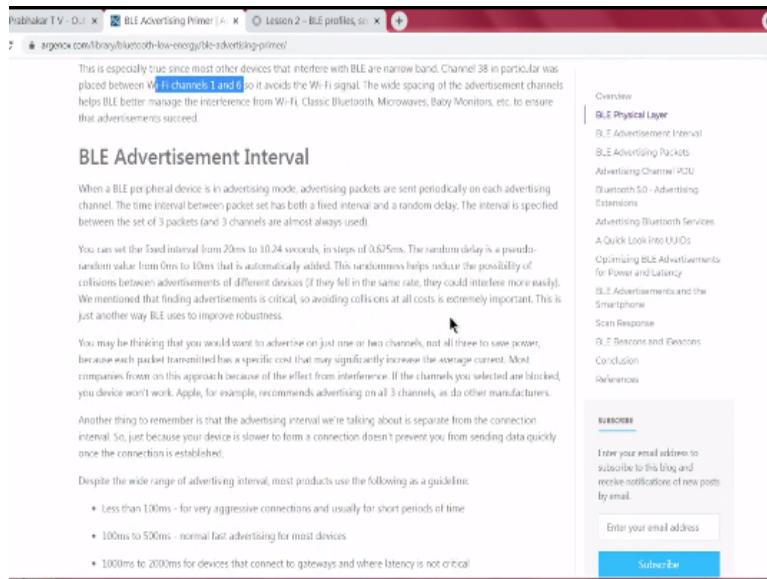
(Refer Slide Time: 06:11)



And you are transmitting your advertising on all the three advertising channels. Bluetooth 5, for instance says hey, why should you do it only on 3, why not we extend this to all the other channels as well. And that is what BLE 5.0 does. It goes beyond the 3 but takes it into many more. But anyway, that is all advanced topics, which if you read you will understand it very soon, okay.

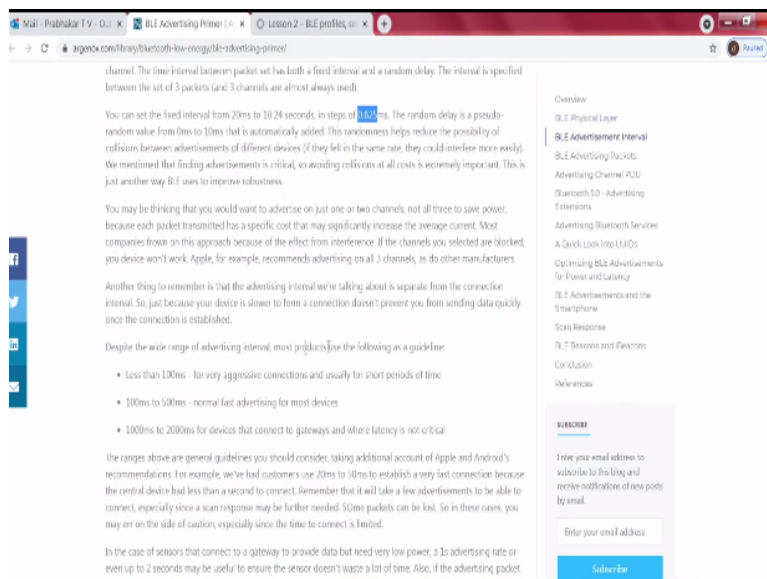
He actually cautions you on 37, 38 and 39 and says about 38. And he says 38, folks you have to be careful because it also maps into the Wi-Fi channels 1 and 6. So it avoids the Wi-Fi signals, right? And it is something that is non-overlapping with Wi-Fi, okay. So the wide spacing of the advertisement channels is good because it can manage interference from Wi-Fi, Classic Bluetooth and microwave, ovens, baby monitors etc. and all that.

(Refer Slide Time: 07:09)



Now this BLE advertisement interval, I think went through this already. So these numbers I already reeled off to you and what is the step size and all that. So you have to think about it when it comes to managing the battery life of these systems.

(Refer Slide Time: 07:26)



Here is a nice number for you to think about, okay. What it says is, if you configure the advertisement interval to less than 100 milliseconds, it becomes very aggressive for connections. And usually for short periods of time, you may want to do that. 100 to 500 is normal, fast advertising for most devices. And 1 second to 2 seconds for devices that connect to gateways and where latency is not critical, okay.

1000 milliseconds to 2000 milliseconds. This is just a number folks. You can use these numbers for mapping it into your own requirement and accordingly configure the advertisement interval using the tools that are available for your embedded node, okay. Then he goes on to say about an example of customers, how they use 20 milliseconds to 50 milliseconds, which is essentially here.

This is that very aggressive connections to establish a very fast connection, that is right. You will be able to do a fast connection because of this. But he also says that it will take few advertisements to be able to connect, especially since scan response may be further needed. I will explain that in a few minutes. It is also possible that some packets may be lost. So in this cases, you may err on the side of caution, especially since the time to connect is very limited.

(Refer Slide Time: 08:53)

• 1000ms to 2000ms for devices that connect to gateways and where latency is not critical

The ranges above are general guidelines you should consider, taking additional account of Apple and Android's recommendations. For example, we've had customers use 20ms to 50ms to establish a **very fast connection** because the central device had less than a second to connect. Remember that it will take a few advertisements to be able to connect, especially since a scan response may be further needed. Some packets can be lost. So in these cases, you may err on the side of caution, especially since the time to connect is limited.

In the case of sensors that connect to a gateway to provide data but need very low power, a 1s advertising rate or even up to 2 seconds may be useful to ensure the sensor doesn't waste a lot of time. Also, if the advertising packet provides data which doesn't change much, then there's no need to transmit frequently.

Make sure to realize that a connection may take a multiple of the advertising interval.

Most devices actually create a more complex system for advertising, using a Fast and Slow advertising regimes. The device boots up or is told to start advertising at a fast rate because the user has interacted with it. This is done for a limited amount of time to provide fast response when the user is expecting a quick connection.

After some pre-determined time, when no connection has occurred, the device then switches to a slow advertising rate which allows an app to connect, but limits the power consumption since the user may take some time to connect.

Advertising rates when a user has already connected depend on whether you want to use directed advertising and user requirements. But since the device may remain unconnected for significant amount of time, it's important to limit it to conserve power.

If your product is intended to connect to iOS devices such as iPhones and iPads, we recommend following the [Accessory Design Guidelines for Apple Devices](#).

Doing so will help speed up device discovery.

Overview
BLE Physical Layer
BLE Advertisement Interval
BLE Advertising Packets
Advertising Channel PDU
Bluetooth SD - Advertising Extensions
Advertising Bluetooth Services
A Quick Look Into UUIDs
Optimizing BLE Advertisements for Power and Latency
BLE Advertisements and the Smartphone
Scan Response
BLE Beacons and Beacons
Conclusion
References

SUBSCRIBE

Enter your email address to subscribe to this blog and receive notifications of new posts by email.

There are issues of connecting fast while you get fast connection. There are also compromises in terms of packet losses, battery power takes a beating, right. All these issues have to be borne in mind. If you are talking about sensors that connect to a gateway, he wants you to put anywhere between 1 second to about 2 second, and says that this is already good enough for you because that is what it is for devices that connect to a gateway where latency is not all that critical.

So you choose that. So he gives you some sort of ballparking on the kind of numbers that you have to use, dividing the full range of advertisement interval into divisions which are most useful for you, okay. Now he also goes on to say that most devices actually create a more complex system of advertising. They use fast and slow advertising regimes. And you can also think about mixing both of them together.

The device boots up and is told to start advertising at a fast rate. And this is done for a limited amount of time and after that, you can expect when the user is expecting a quick connection you do this. After some predetermined time, when no connection has occurred, you can switch back to slow advertising.

That means on the fly, you should be able to go from fast to slow so that you do not lose out on performance at the same time, you will be able to save your battery life, it does not explode too much. So that is another important thing.

(Refer Slide Time: 10:28)

Advertising rates when a user has already connected depend on whether you want to use directed advertising and user requirements. But since the device may remain unconnected for significant amount of time, it's important to limit it to conserve power.

If your product is intended to connect to iOS devices such as iPhones and iPads, we recommend following the Accessory Design Guidelines for Apple Devices. Doing so will help speed up device discovery.

BLE Advertising Packets

The Bluetooth Specification defines the top level packet in Bluetooth LE with two data units. The packet itself has several parts including a preamble and access address, as well as a CRC.

The Packet data unit for the advertising channel (called the Advertising Channel PDU) includes a 2-byte header and a variable payload from 6 to 37 bytes. The actual length of the payload is defined by the E-bit Length field in the header of the **Advertising Channel PDU**.

It's important to note that there are several PDU types for the advertisements, but here we will focus primarily on **ADV_IND** and **ADV_NONCONN_IND**.

ADV_IND is a generic advertisement and usually the most common. It's generic in that it is not directed and it is connectable, meaning that a central device can connect to the peripheral that is advertising, and it is not directed towards a particular Central device.

When a peripheral device sends an **ADV_IND** advertisements, it is helping Central devices such as Smartphones find it. Once found, a Central device can begin the connection process.

ADV_NONCONN_IND is the advertisement type used when the peripheral does not want to accept connections, which is typical in Beacons.

The right Advertisement filter to use depends on your application, whether you want to form a quick connection or

Overview
BLE Physical Layer
BLE Advertisement Interval
BLE Advertising Packets
Advertising Channel PDU
Bluetooth LE - Advertising Extensions
Advertising Bluetooth Services
A Quick Look into UUIDs
Optimizing BLE Advertisements for Power and Latency
BLE Advertisements and the Smartphone
Scan Response
BLE Beacons and Beacons
Conclusion
References

SUBSCRIBE

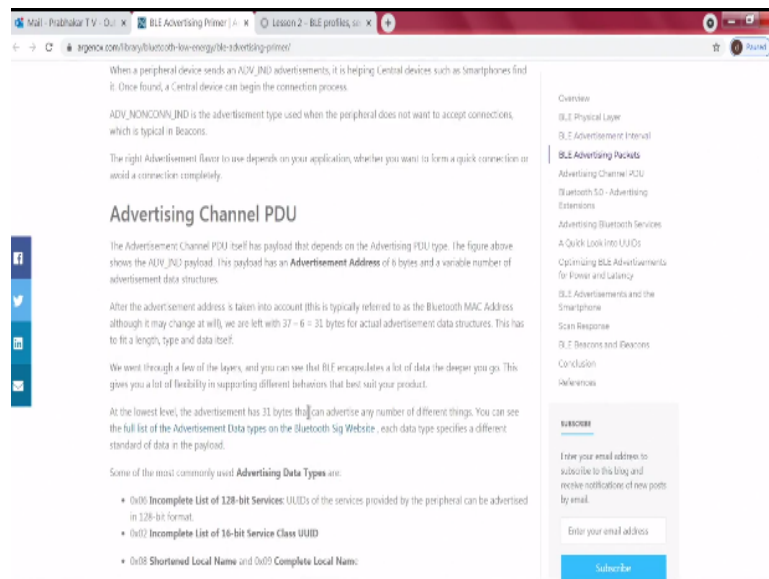
Enter your email address to subscribe to this blog and receive notifications of new posts by email.

Subscribe

Then he talks about BLE advertisement advertising packets. Basically, he talks about two types of packets. One is **ADV_IND** and the other is the **ADV_NONCONN_IND**. Now this name looks very familiar. **NONCONN_IND** simply stands for, it is a advertisement type but it does not want to accept connections. That is why if you send an advertisement packet as **ADV_NONCONN_IND**, it is just nothing but a beacon packet.

So central devices will not make an attempt to connect back to this device because they know that this is the type of advertisement packet. It is ADV_NONCONN_IND. However, if you want to advertisement you want to send advertisement for the purposes of connection then you have to call it ADV_IND and this is a generic advertisement which you essentially use.

(Refer Slide Time: 11:31)



Then the article talks about what is the size of the advertising channel PDU, what is the size of the PDU. Now see the advertisement address itself is six bytes and there is a variable number of advertisement data structures, okay. Now what he wants you to do is he says after the advertisement address is taken into account we are left with you have basically 37 bytes.

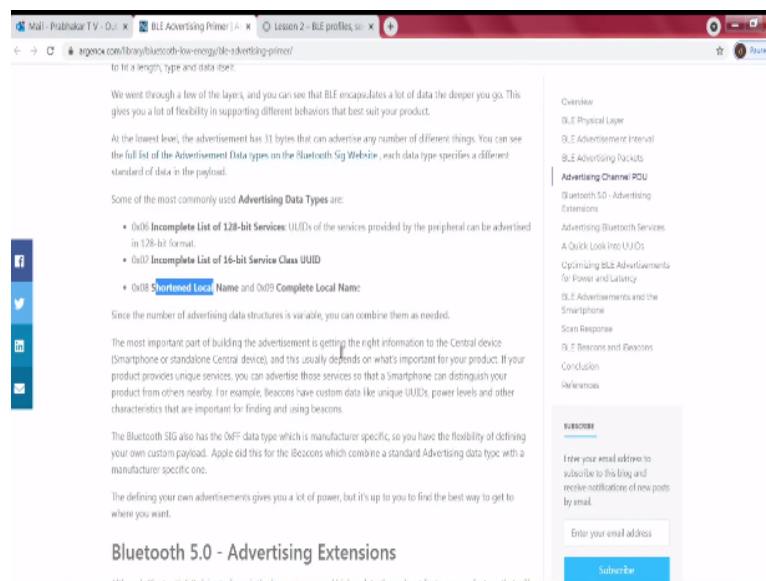
Out of that 6 bytes go away for advertisement address so which leaves you with 31 bytes, okay. Now how is this 31 bytes, what can what can you do with this 31 bytes? Well out of this 31 bytes, some of the bits are actually gone. Depending on what you want to do in the remaining 31 bytes you will have to think about what are the advertising data types that you want to put into the payload.

For example, if you put x06 this is nothing but incomplete list of 128 bit services which the BLE is offering. UUID services provided by the peripheral can be

advertised in this 128 bit format. And 02 if you put it is completely stop service class UUIDs. And if you put 0x8 it is shortened local name. And if you put 9 it is complete local name, okay.

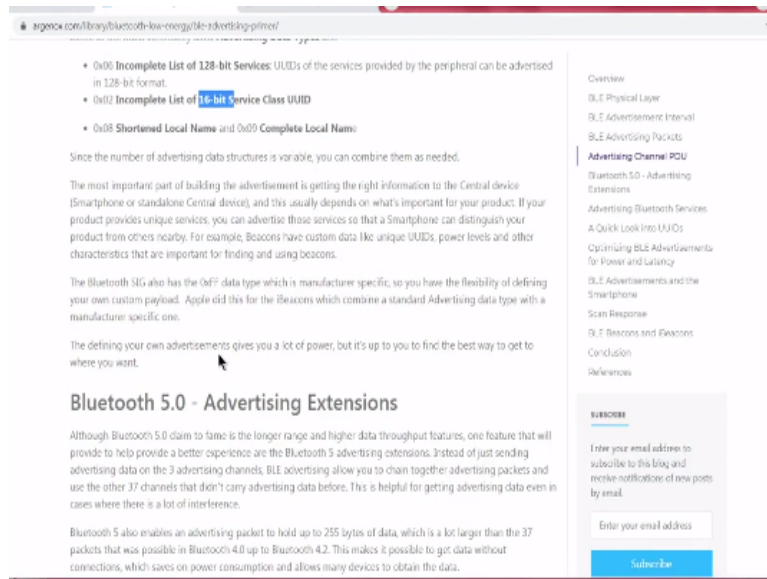
So these are the different types of advertising data types which are most commonly used, okay. You can go and look up the structures even more. You will realize that there are much more information what we are actually discussing here.

(Refer Slide Time: 13:15)



Now so you can see this 31 bytes, out of 31 bytes already depending on what you are putting into the payload, the advertising data type already takes away 128 bit and 16 bit and as you can see that is additional 16 bytes there. Okay, here is 16 bytes, which is just going away. And this is 2 bytes and this is some other type. So you can see that this 31 bytes, out of 31 bytes how it is, already the payload is getting hogged by this advertising data types, okay.

(Refer Slide Time: 13:54)

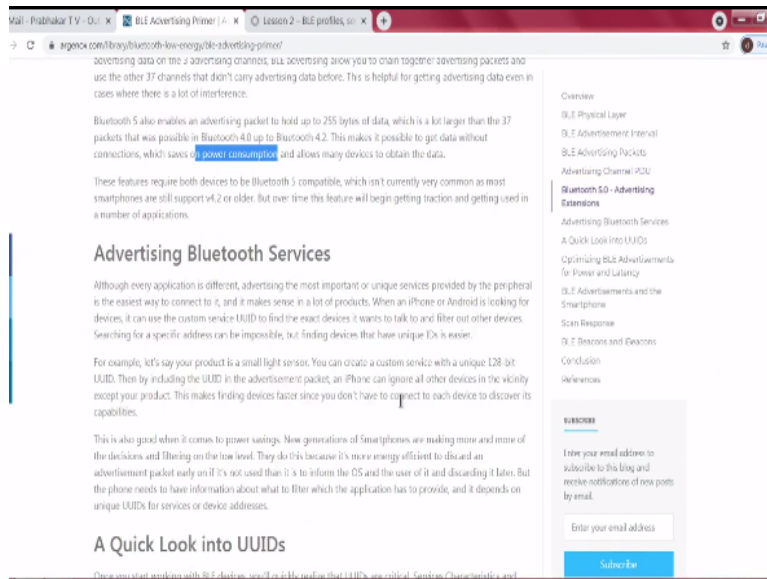


You can also define your own advertising types and it gives you a lot of power. But it is up to you to find the best way. Now I mentioned to you that instead of going on just the three advertising channels, you can also do advertising extensions on other channels.

And also useful is the fact that you are not no short of any space because out of 31 bytes 16 bytes are already gone depending on the type of advertising data type, you can actually go up to 256 by 255 bytes of data which is a lot larger than the 37 – 6 which was out there. So you will be able to make, you will be able to advertise more data without making connections.

And therefore you will be able to save power yet push a lot of data back to the central without having the overhead of connection. So this is attractive from that side of the perspective because it is indeed energy efficient quite a bit.

(Refer Slide Time: 15:01)



Now advertising Bluetooth services, you may want to look up you UUIDs because that is where the different profiles and different services which are there are clearly defined there. And we will go into the details of that. But before that, there are Bluetooth services which iPhone and Android are looking for devices which are trying to connect to them. So every application is different.

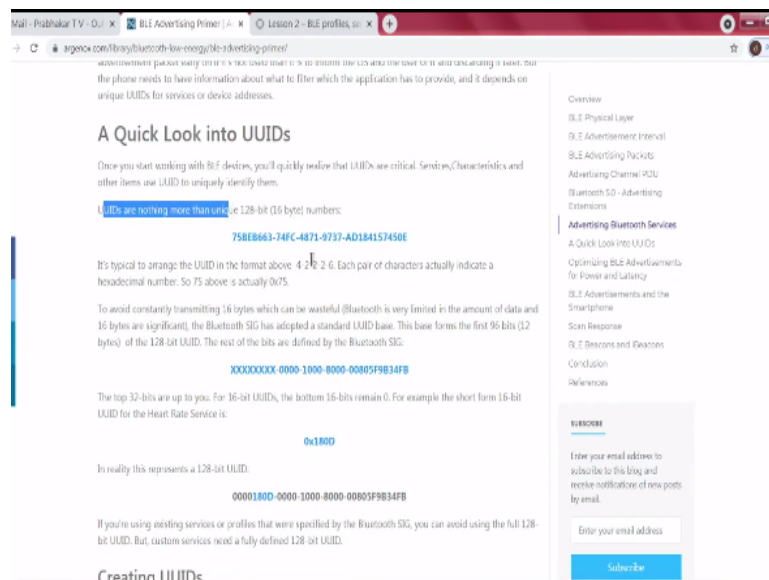
So advertising is most important and unique or unique services provided by the peripheral is the easiest way. So phones are essentially the ones that humans are holding on to and you may want to and the nodes which are trying to connect are essentially the ones that are offering important and unique services because they are peripherals.

So it is important to carry as much information in the advertisement packet about the kind of services which these peripheral nodes can offer. Yet at the same time, you have to ensure that it comes with a good amount of power savings. And smartphones are making more and more of the distance or filtering on the low level. For instance, this is a nice thing because let us say smartphones are interested only in few types of peripherals, okay.

There it can actually put a filter, okay. And this is much more energy efficient by just putting that filter you read only from a subset of UUIDs use only read from a subset

of peripherals, which are associated with UUIDs. I will explain that in a moment. So filtering is important again from a power savings perspective. And that is what this article is actually mentioning about what you can do at the lower level.

(Refer Slide Time: 16:43)



Now what are these UUIDs? These are nothing but numbers folks as he says you know you can see this is nothing but a number. So this 128 bit or 16 byte is nothing but a number as long as this number here, okay. It is typical to arrange the UUID in some format. It is 4-2-2-2-6, okay. This is how your byte sequence goes and you can see 4 and 2 is 6, 6 and 4 is 10. So this is 16 byte and this how it is divided.

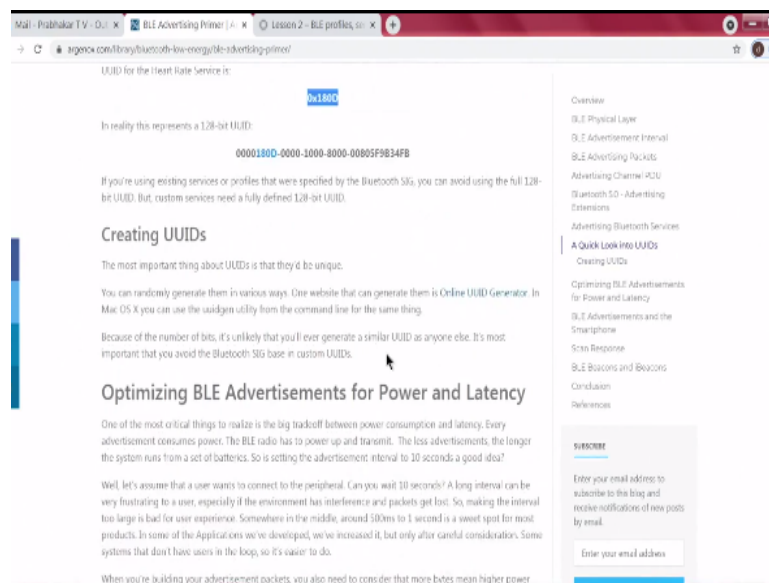
Each pair of characters actually indicate a hexadecimal number. So you can think of an example like this. 75 is actually anything but x75. You take this number here, this is x75. Then you have this xBe and so on and so forth. So it is just one way of one notation there. So but the point is that if you want to constantly avoid transmitting the 16 bytes, which can be wasteful the Bluetooth SIG has adopted a standard UUID base, okay.

There is one database in which all these UUIDs are entered. This base forms the first 96 bits. You do not have to transmit all of them. The base of the first 96 bits are already out there, okay. So 12 bytes are already out there, out of the 16 bytes. The rest

of the bits are defined by the Bluetooth SIG and that top 32 bits are up to you. For 16 bit UUIDs the bottom 16 bits remain 0.

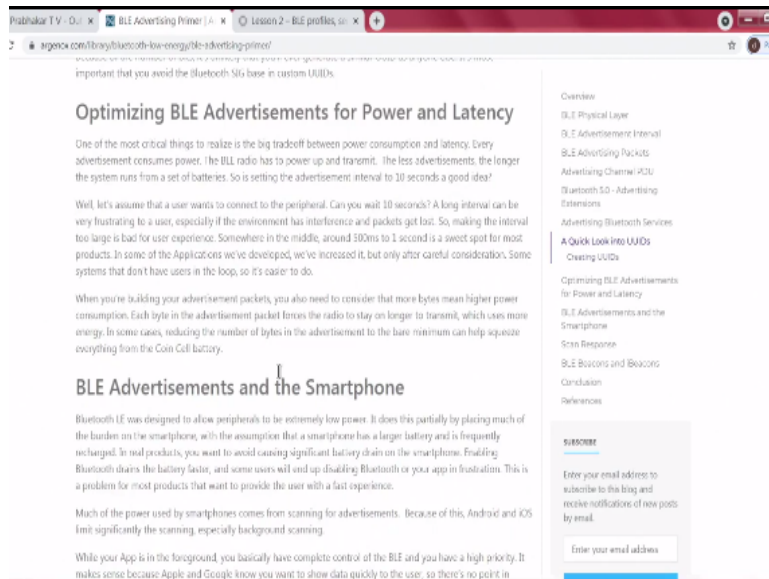
For example the short form 16 bit UUID for the Heart Rate Service is this. It is mentioned here. So depending on the advertisement type, data type, you can see that you will be able to configure, you will be able to use the payload in an effective manner. Essentially that is what this article is trying to get at, okay.

(Refer Slide Time: 18:47)



You can also create your own UUIDs and only thing is they need to be unique. You can go to one UUID online generator and obtain that 128 bit number there and then use it for your communication. Put filter so that you can only receive from a set of 128 bit UUIDs so that you can effectively save on power. All right.

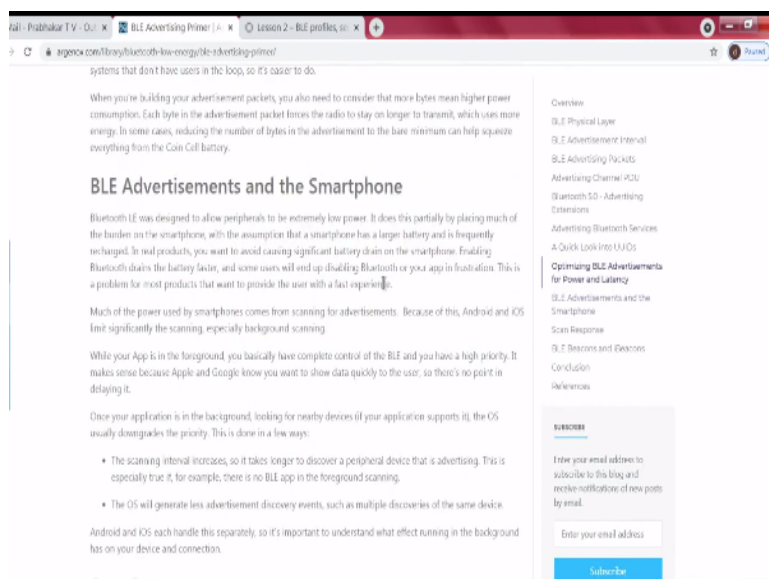
(Refer Slide Time: 19:12)



So optimizing BLEs for power and latency I think we discussed this many times. Again there is a nice paragraph on the, you know in terms of the advertisement intervals. So you look at this. He says how long can you wait. Is it 10 seconds or is it a little longer. So take your decision based on that. So he also gives you a number of 500 to 1 second if you set what will happen.

If you set it to 10 seconds what is going to be happening. It will be very frustrating for user because you send out something and you have to wait for a significantly amount of time. So all that is mentioned in this paragraph, okay.

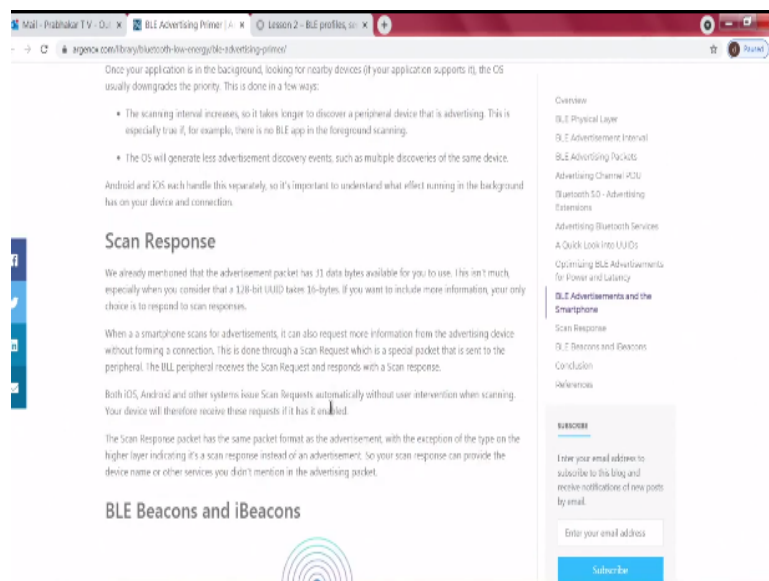
(Refer Slide Time: 19:55)



So another important aspect is about BLE advertisements. This paragraph is about BLE advertisements and the smartphone. You as I mentioned to you, much of the power used by the smart phones is coming from scanning for advertisement. See, thing is you can do, basically the phone is the central, right? And therefore, it has to be on and it has to look for peripherals and so on.

So scanning is a very important aspect in the phone. And that scanning can be power consuming. So what you can do is you can limit the amount of scanning and put it into as a background scanning mode, okay. But that is a little bit frustrating because devices will take a little longer to connect. However, you will be doing that well because the phone energy battery energy will be conserved quite a bit if you put it into background scanning. So this is telling you about that aspect of the whole system.

(Refer Slide Time: 21:00)



Here is another paragraph about scanned response. See folks, I mentioned to you about the fact that the payload will give you only $37 - 6$, that gives you 31 bytes of data. But this is not all that big because if you subtract 16 bytes already which is the 128 bit UUID, you are hardly left with anything there right. So which is in fact you are left only with 15 bytes.

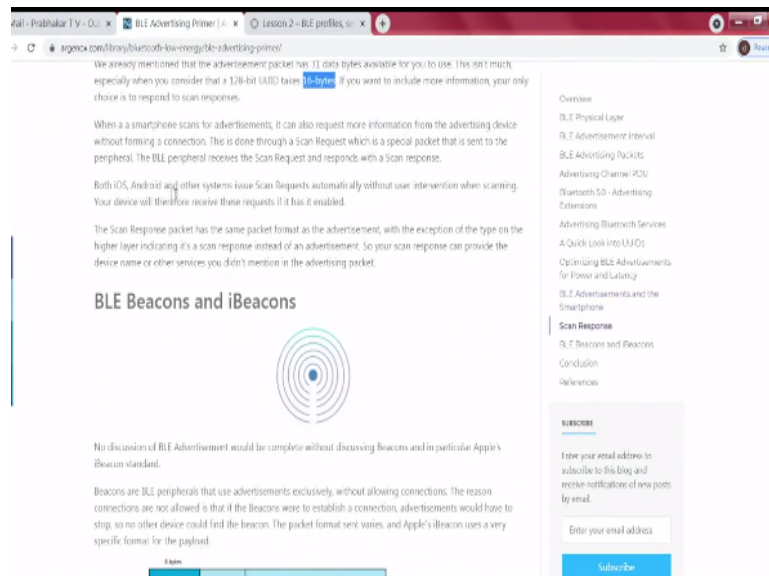
And that may be insufficient for you to convey all the kind of, all services that you are you have with you. Therefore, the BLE protocol actually allows you to do something

clever. It says I will give you, the central can do a scan request and in return the peripheral can do a scanned response. And that scanned response can actually carry the remaining part of the information which you could not have conveyed by your advertisement packets.

That is the key point. If you send advertisement packets, you can only send whatever 31, you will be able to send only 15 bytes right, because 31 bytes has already gone away for UUID. You are left with 15 bytes. Your 15 bytes may not be sufficient to convey all the information and repeatedly you have to keep sending advertisement packets. So anything greater than 15 byte is not possible.

Therefore, the standard says the central can invoke a scan request for which the peripheral can do a scan response and add that additional information which the peripheral wanted to convey which should not have been possible without establishing a connection. So this is essentially what scan response is, okay.


(Refer Slide Time: 22:51)

A screenshot of a web browser displaying a page titled "BLE Beacons and iBeacons". The page content includes text explaining that a scan response packet has the same format as an advertisement packet but with a different type on the higher layer. It also mentions that both iOS and Android automatically issue scan requests. A diagram shows concentric circles representing a beacon's range, with a "6 km" label. A sidebar on the right lists navigation links such as "Overview", "BLE Physical Layer", and "Scan Response". At the bottom right, there is a "SUBSCRIBE" section with an email input field and a "Subscribe" button.

And both iOS, Android and other systems actually support scan request automatically without user intervention while scanning.

(Refer Slide Time: 23:01)

BLE Beacons and iBeacons



No discussion of BLE Advertisement would be complete without discussing Beacons and in particular Apple's Beacon standard.

Beacons are BLE peripherals that use advertisements exclusively, without allowing connections. The reason connections are not allowed is that if the Beacons were to establish a connection, advertisements would have to stop, so no other device could find the beacon. The packet format varies, and Apple's iBeacon uses a very specific format for the payload.

8 bits											
MAC Address			AD 0		AD 1						
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B
0x0C	0x0D	0x0E	0x0F	0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17
2 bytes	1 byte	1 byte	14 bytes	14 bytes	14 bytes	14 bytes	14 bytes	14 bytes	14 bytes	14 bytes	14 bytes

You can see the format for the iBeacon packets. These packets use the basic BLE format, with some specific fields. Let's go through them one by one.

The advertisement packet contains the Bluetooth MAC address and the payload. The payload is composed of two AD Structures, the first one gives generic information using the Flags Data Type, and the second is the Apple-specific Beacon information.

Now no discussion on advertisements, BLE advertisement is possible if you do not know about BLE Beacons and iBeacons. So folks, I encourage you to read this. This is as I told you, this is like a simplex connection, it is a beacon and Apple has defined the frame structure if you want to receive a beacon BLE Beacon from an apple onto an Apple phone there is something called the iBeacon standard.

So this is the iBeacon standard here as you can see, this whole thing. And this will give you only advertisement and no allowing of any connections. This is important and quite like that there is EddyStone which is another format, okay.

(Refer Slide Time: 23:46)

You can see the format for the iBeacon packets. These packets use the basic BLE format, with some specific fields. Let's go through them one by one.

The advertisement packet contains the Bluetooth MAC address and the payload. The payload is composed of two AD Structures, the first one gives generic information using the Flags Data Type, and the second is the Apple-specific Beacon information.

Flags Advertising Data Type

This packet has data type 0x01 indicating various flags. The length is 2 because there are two bytes, the data type and the actual flag value. The flag value has several bits indicating the capabilities of the iBeacon:

- Bit 0 - Indicates LE Limited Discoverable Mode
- Bit 1 - Indicates LE General Discoverable Mode
- Bit 2 - Indicates whether BR/EDR is supported. This is used if your iBeacon is Dual Mode device
- Bit 3 - Indicates whether LE and BR/EDR Controller operates simultaneously
- Bit 4 - Indicates whether LE and BR/EDR Host operates simultaneously

Most iBeacons are single mode devices BR/EDR is not used. For iBeacons, General discoverability mode is used.

iBeacon Data Type

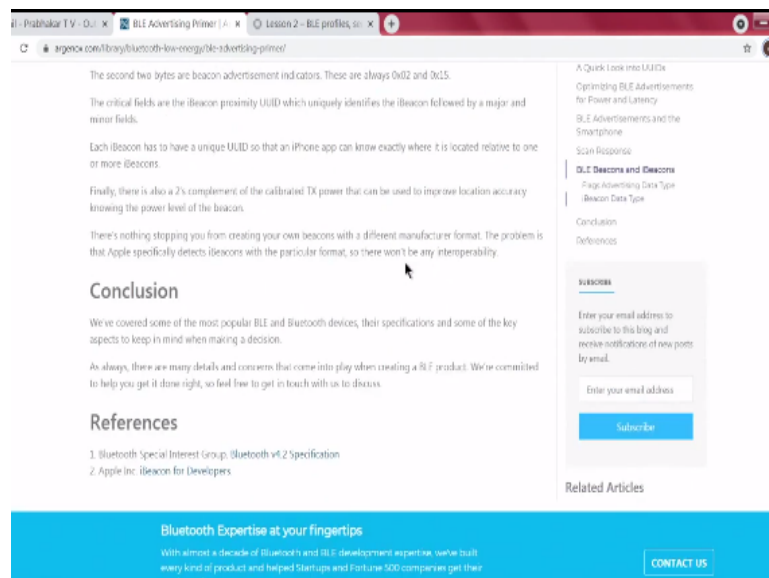
The most important advertisement data type is the second one. The first byte indicates the number of bytes, BCAA for a total of 25 bytes, 25 for payload and one for the type. The AD type is the Manufacturer Specific, 0xFF, so Apple has defined their own Advertisement Data.

The first two bytes indicate the company identifier (0x4C00). You can see identifiers for other companies as well.

The second two bytes are beacon advertisement indicators. These are always 0x02 and 0x15.

Now there are flags associated with the advertising data type, you can read up these things here. And iBeacon data type is also mentioned in quite a bit of detail here. So what is the big summary?

(Refer Slide Time: 24:01)



Summary is the BLE and Bluetooth devices particularly the advertisement intervals have to be looked at a little more carefully because there is a wealth of information about how you can build interesting applications with advertisement packets alone without really connecting to any device, which today may not be possible if there is no security.

Why should anyone allow you to you know connect to your system? Therefore advertisement is a good way to go, right way to go. Read up this article, understand about UUIDs, okay and all the power related issues which you will be able to build a very interesting applications with BLE in the advertisement more. Thank you very much.