Questions

#todo

- Why in LE example why D is advertiser and A scanner, but not the opposite?
- What if in Scatternet they use the same frequencies? Do they overlap?
- What Logical transport types is the Channel access BR EDR example

Table of contents

Bluetooth

#card

- Bluetooth is a **collection of protocols** that are a **cable replacement technology** (pc, headphones, printers) that implement a personal area network
    - Lower cost and lower power consumption than IEEE 802.11
- It was more successful with smartphones
- Limited coverage (one room) up to 10 meters
- Slow versions because it's not a replacement of Wi-Fi, nowadays high speed versions uses Wi-Fi inside

Versions

#card

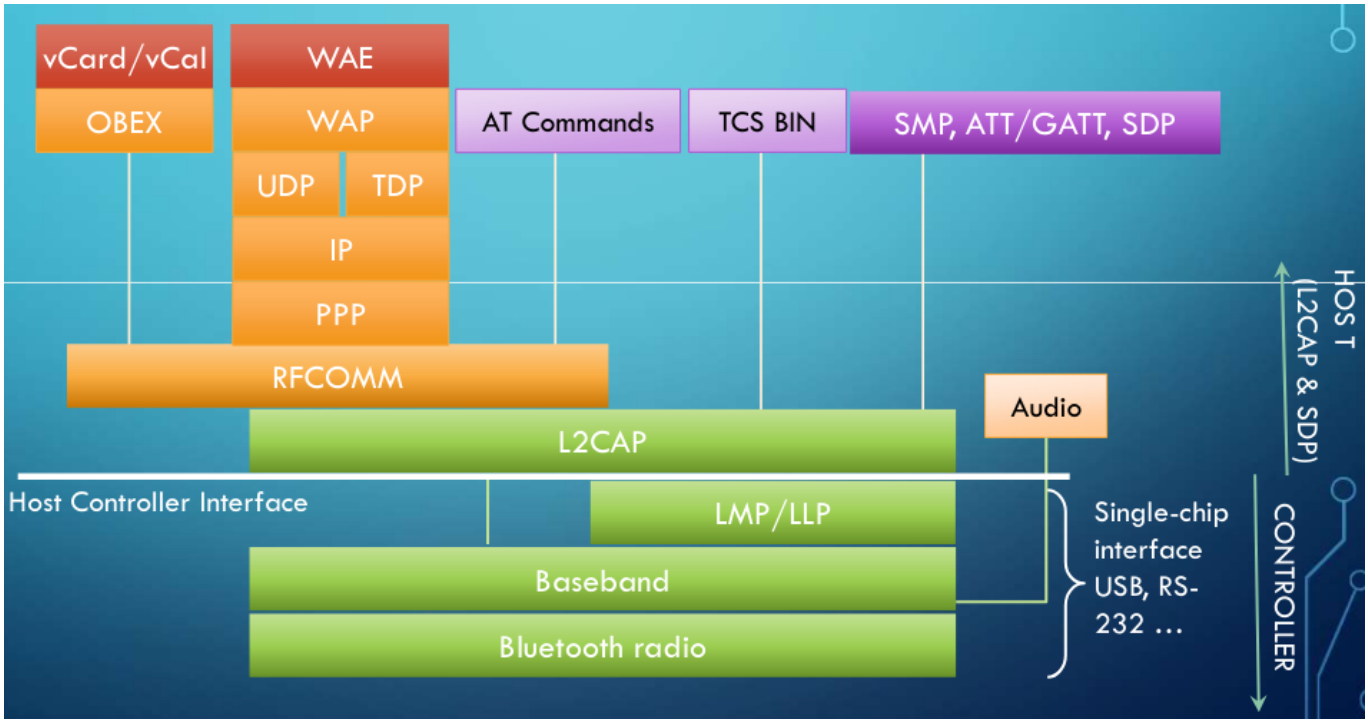
- 3.0, uses a parallel IEEE 802.11 channel to increase up to 24 Mbps, that is negotiated with Bluetooth
- 4.0 in 2010, defines: classic Bluetooth, high speed Bluetooth and low energy Bluetooth (i.e. BLE ~1 Mbps)
- 5.0 in 2016, BLE with burst up to 2Mbps (important for IoT) and increase capacity for connectionless services (location relevant navigation)

Applications
#card

- cable replacement (original idea)
- sync devices (e.g. smartbands)
- access to the internet
- mobile social networking (e.g. Immuni or crowd sensing in general)

Architecture

#card



- Host controller interface between hw/sw
- **One host** (logical layer), while **one or more controllers** for the physical layer for the different bluetooth versions
    - A bluetooth card contains the controller + L2CAP (logical link control and adaption protocol)
- Protocols (core in **bold**, others are adopted)
- **Physical layer** defines

    - **Radio**, frequencies
    - **Baseband**, low level procedures
- **Data Link layer** defines the **setup and controller**
    - Lower part
        - **LMP** (link management protocol for BR/EDR)
        - **LLP** (link layer protocol for BLE)
    - Upper part
        - **L2CAP** (Logical Link Control and Adaptation): higher-level protocol multiplexing (package segmentation)
- **Application layer**
    - Telephony AT, TCS
    - Management/discovery
        - **SDP** for service discovery
        - SMP, ATT/GATT (kind of a 1small ZigBee cluster library)
    - RFCOMM emulates serial port for cable replacement

Controllers

#card

- Basic rate (headphones for smartphone)
- Low Energy
- Alternate MAC, for high speed transport link IEEE 802.11 (2 Mbps → 24 Mbps)

Basic rate or Enhanced data rate

#card

- Operates at 2.4Ghz like IEEE 802.15.4 and IEEE 802.11
- 1Mbps for BR, 3Mbps for EDR
- Forms a **piconet**: group of devices synchronized to a common clock and **frequency hopping pattern** #todo
    - Bluetooth BR/EDR is **connection oriented**, so piconet must be formed before, in order to communicate
- Logical links transport **synchronous** (like IEEE 802.15.4), **asynchronous** and **isochronous** data (start async and then for a period sync)
    - Synchronization is used to ensure low energy for BLE, because it's required at mac level and using sniff mode
- BR uses at peak 25mA

Low Energy (LE)

#card

- Compared to Basic rate Enhanced data rate
  - smaller packets to reduce TX/RX power consumption
  - less channels for discovery and connection (searching over a smaller n)
  - simpler state machine
  - used to **expose the state** rather than transferring
- Frequency division multiple access (FDMA) over 40 physical channels + Time division multiple access (TDMA).
- Physical channel divided with **advertising** and **connection** events, namely frame and its response
- other
  - latency 3ms
  - topology star
  - connections > 2billion
  - range 150m

Powe consumption

#card

- less than 20mA at peak with an average of less than 5µA
  - even devices with coin cell battery!

Speed

#card

- up to 1Mbps nominal, but in reality for the overhead it's smaller

Topology

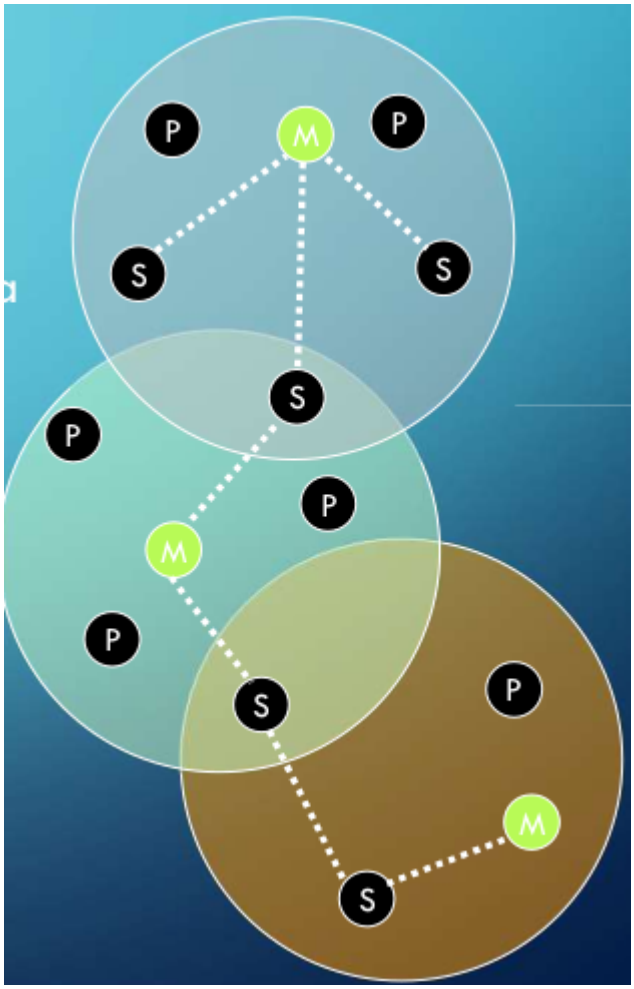
#card
It's the piconet, formed by master and slaves

- **1 Master** than enforces the **synchronization** to enable the communication of the slaves, by **controlling the access** to the channels of the **slaves**
  - roles are not fixed
  - hence slaves can only communicate when the master says so
- Types
  - Point to point
  - Point to multipoint
- BR/EDR

    - Up to 7 slaves in a piconet, but other can be in **parked mode** to keep in sync with the master.
- LE
  - No limit to active slaves in a piconet
  - Just in one piconet

Scatternet

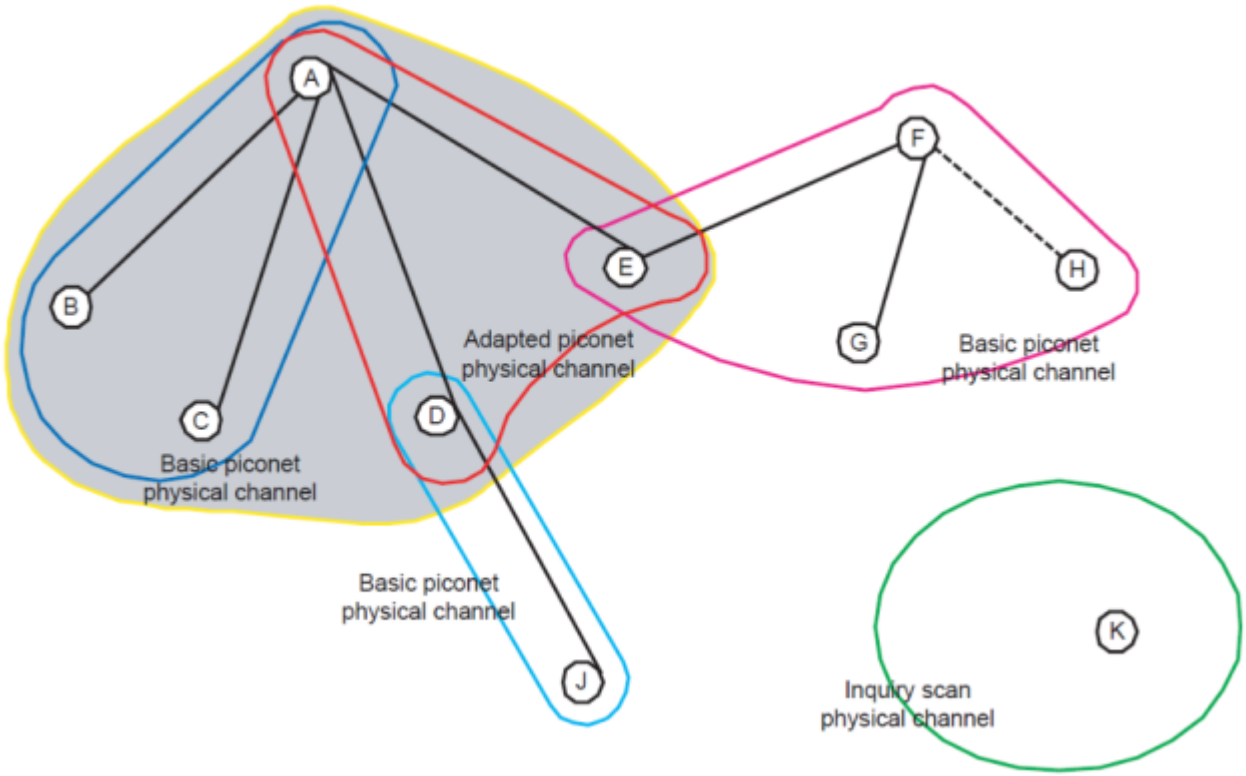
#card



- piconets share the same frequencies
- to extend the range by doing multi-hop communication
- there is no routing protocol defined by the standard
- no useful applications

BR/EDR example

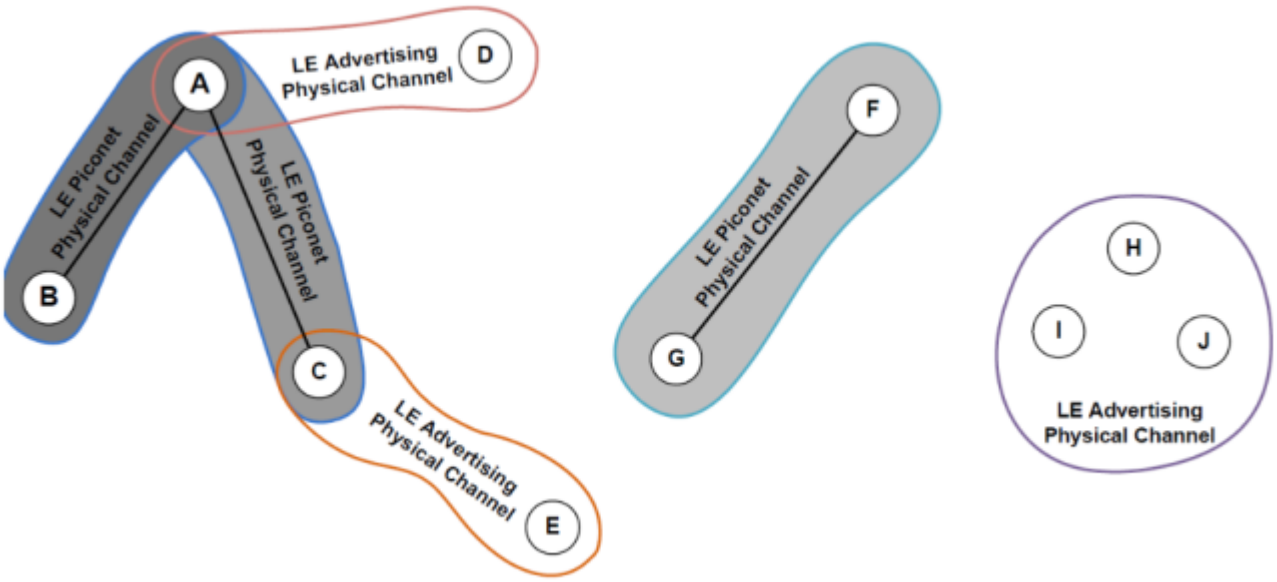
#card #exam



- Several piconets in an area
- A is the master, the other slaves
  - They can use different low level mechanism
- F is master, the other slaves
  - H is temporarily **deep low power mode** maybe using the sniff state
- E is a slave both in the piconet of A and F, so it need to keep in sync with both in different time
- D is a slave in A but a master in his piconet with J
- K is not connected to any piconet but scanning for it in order to communicate
- Slaves cannot communicate point to point, but need to pass through the master

LE example

#card #exam



- communication
  - connection-oriented channels (grey one)
  - acts like the master in BR/EDR, so the master A will enforce the sync of the slaves B and C by telling them when to communicate
  - connection-less advertising channels, transparent
  - D is an **advertiser**, that is not being connected to a piconet but emitting periodically beacons to expose the state to any device in range that is scanning, in this case A is a **scanner**.
  - Be aware that D is not a slave, because there is no connection
  - I, H, J are not in a piconet, but they are advertising/scanners or just one of it.

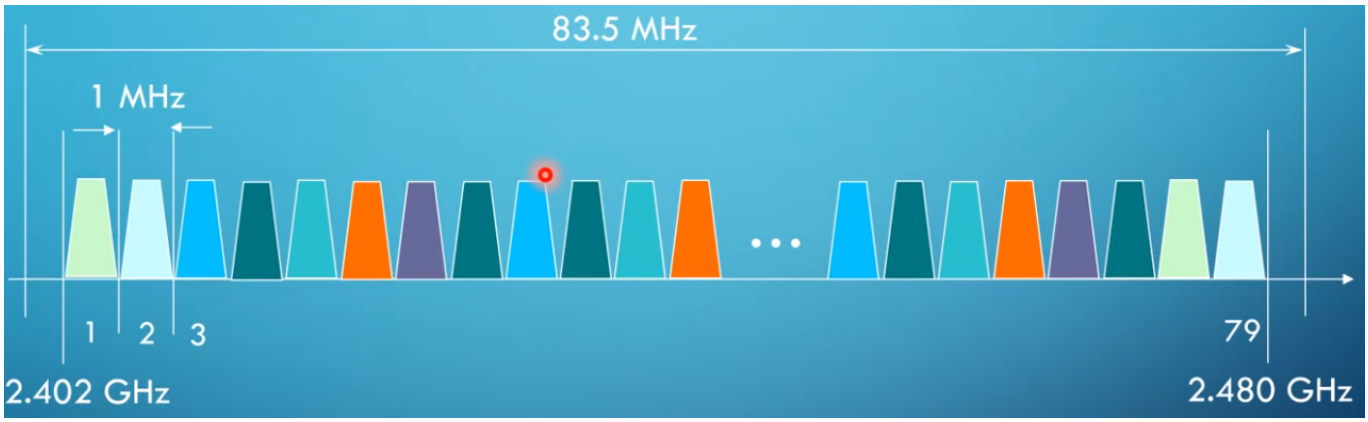
Layers

Bluetooth radio layer (physical)

#card
defines the specification of the **radio frequencies**

BR/EDR

#card



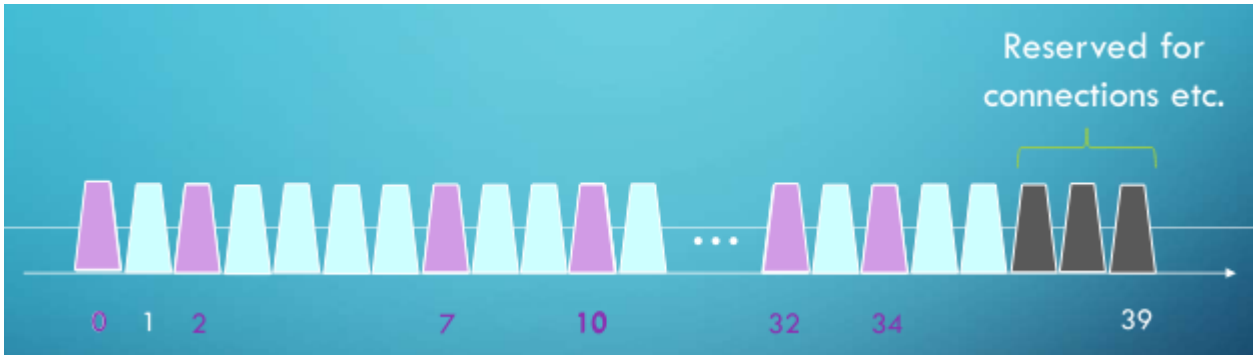
- it operates in the 2.4GHz **frequency band** with a range of 83 MHz
  physical channel (synchronized to a common clock and hopping sequence).
- the frequency band is divided in sub-bands of 1MHz, obtaining 79 **physical channels**
- the master defines a pseudo-random sequence
- every time a frame needs to be transmitted, the next channel is computed (**hopping**) and the radio is set to transfer over that frequency band (channel).
  - there are 1600 hops/seconds

Why it doesn't conflict with ZigBee?
#card #question

- sub-bands in ZigBee are just 16, wider, and it stays always on the same channel
- if overlap with Bluetooth, there is change that one of the network is in low-power mode
- different encoding, so noise it's observed and then discarded

LE

#card



- 40 channels
  - frequency hopping only in the first 37 channels used for **connection oriented** (master/slave)

- 37, 38, 39 for connection, discovery and broadcast (search O(3) rather than O(40)), used by **advertisers** to emit beacon and by **scanners** for listening

Transmission power

#low-priority

- class 1: Tx power < 20 dBm (100 mW)
    - Long range (100 m);
- class 2: Tx power < 4 dBm (2.5 mW)
    - Normal range (10 m);
- class 3: Tx power < 0 dBm (1 mW)
    - Short range (10 cm).

Baseband layer (physical)

#card #low-priority
It creates the **abstraction of physical channel** that is the pseudo-random sequence, each piconet with a different sequence

- the slave connects and receive the sequence from the master that syncs the hop (frequencies) and the time
- Frequency hopping management (FH).
- management of the channel and of the link (to LMP/LLP)
    - Power control, Link control (packets retransmission, sync/async link)
    - **Channel access** by using **time division duplex (TDD)**
- Error correction

Logical transport types

#card #low-priority
They are **abstraction over the physical channel**

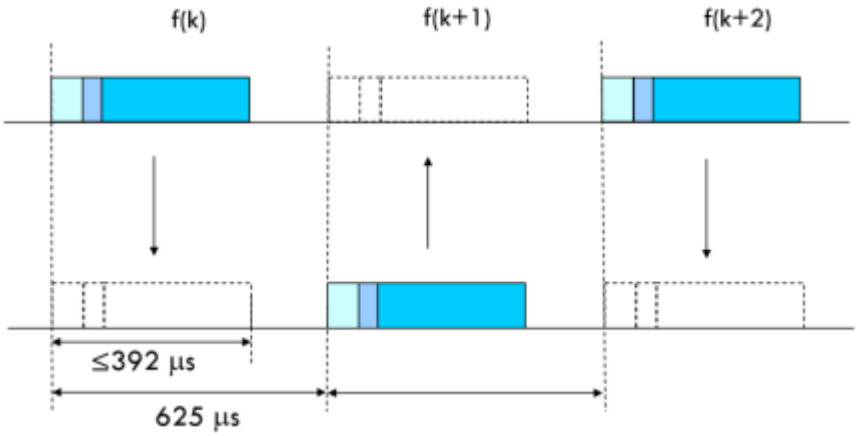
- Asynchronous Connection-Oriented (**ACL**)
    - BR/EDR or LE if piconet
    - a slave can respond or not
- Advertising broadcast (**ADVB**)
    - LE for advertisers
    - there are no ACK, so **unreliable** and **unidirectional**
- Synchronous Connection-Oriented (**SCO**)
    - BR/EDR
    - slots are reserved for the master/slave communication (e.g. audio stream)
- Extended Synchronous Connection-Oriented (**eSCO**)

- Broadcast, can be used to sync the slaves
  - Active Slave Broadcast (**ASB**)
  - Parked Slave Broadcast (**PSB**)

BR/EDR

Channel access

#card #exam



- channel divided with TDD in slots of 625µs each
- master m uses odd slots, slave uses even slots for the master/slave communication
- master m chooses the frequency $f(k)$ in the pseudo-random sequence generated previously
- m transmit first and the slave $s1$ receives in a time-slot of 625µs but the packet duration is less than 392µs, because the remaining is used to switch the frequency of the channel
- $s1$ only can respond in $f(k+1)$
- in $f(k+2)$, it could be that m sends to $s1$ or even another s_n

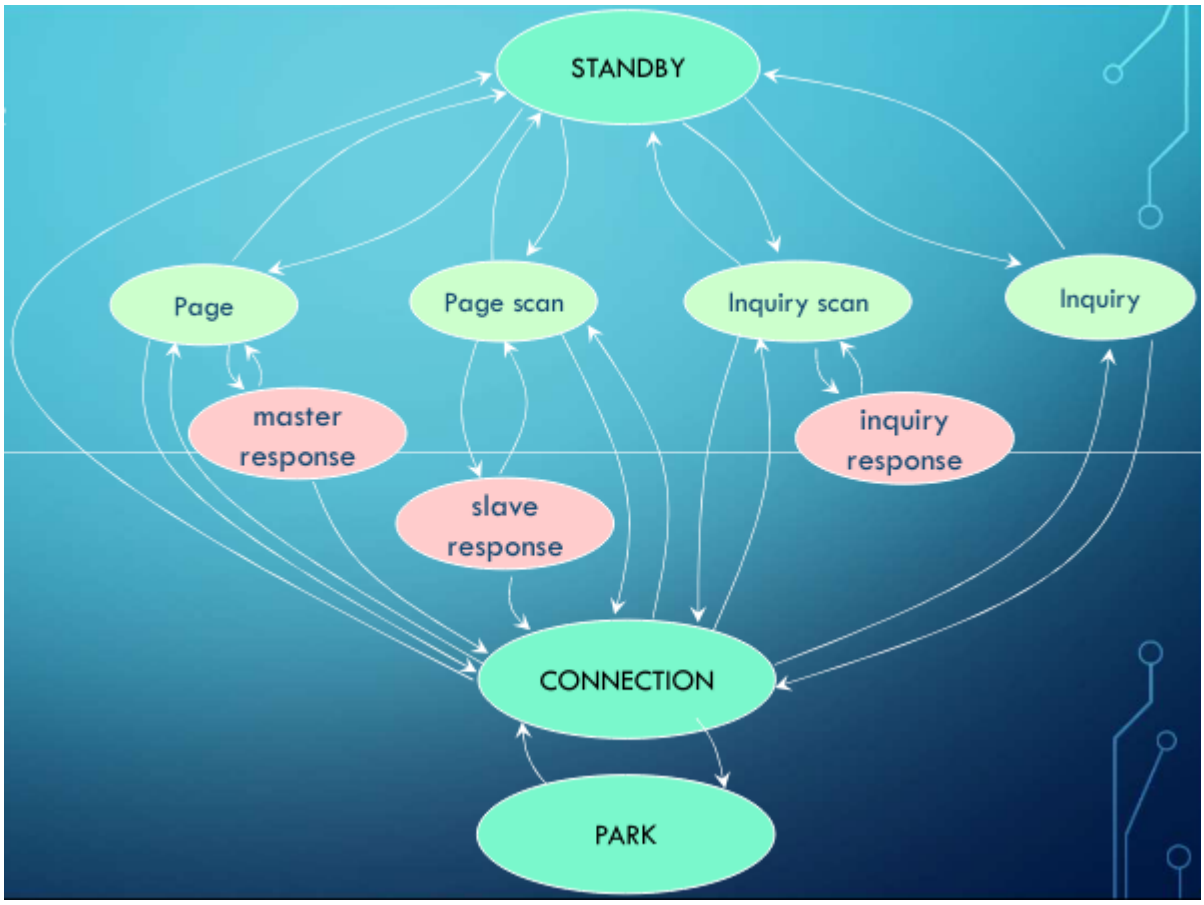
States

#card
Bluetooth is organized as a state machine. There several sub-states, but the major one are where the devices spent most of the time:

- **STANDBY**
  - It is the initial state of devices, not yet in the piconet
  - may be in low-power mode
- **CONNECTION**
  - in the piconet, so now master or slave and can exchange packets
  - may go in low-power mode

- **PARK** (the device is in the piconet)
  - device is in the piconet, but only remain synchronized to the channel (keep receiving broadcasts)
  - Allow a very low duty cycle

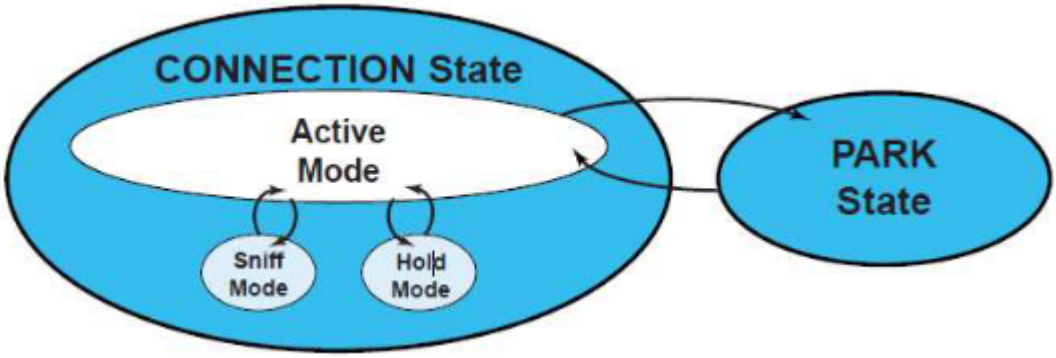
Transitions among the states
#card #exam



- page, inquiry used to implement dynamic connection to the network
- STANDBY → *Page scan*, if a slave is searching for a piconet
- STANDBY → *Page*, if a master is available to receive connections for a piconet
  - looks for devices in *Page scan*, when they detect each other the master switches to *master response*, the slave switches to *slave response*, where the master sends the parameters (clock, hopping sequence)
  - then they switch to CONNECTION where they communicate
- CONNECTION ↔ PARKED, can be done by the salve sometimes
- The discovery of devices can be implemented with
  - STANDBY → *Inquiry*, discover other devices is the piconet by sending messages
  - STANDBY → *Inquiry scan*, if a device want to be discovered by listening to *inquiry* messages. It can optionally answer by sending back device parameters (address, name, supported services...)

Connection state sub-states

#card



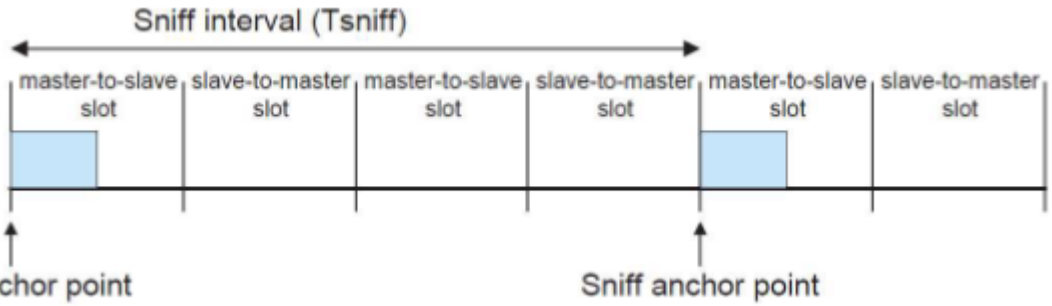
-

- Active mode, normal master/slave communication where the slave can optionally go to PARK STATE
- Switching between active, sniff and hold is fast because address is the same

Sniff mode

#card #exam

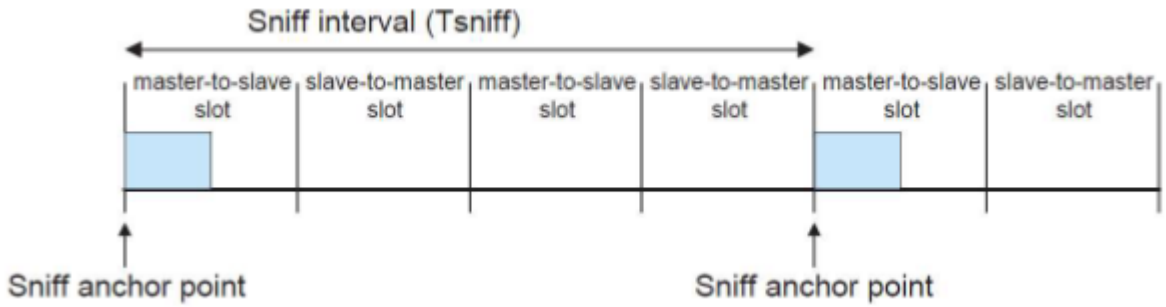
- to reduce the duty cycle to save energy. The slave tells the master it will communicate only in a given specific slots that are smaller than all the available slots.
- it supports both sync/async



- duty cycle of the activate state is represented by all the slots divided with black horizontal line
  - slave require to switch to sniff mode and agrees with the master a **sniff interval** divided by the **sniff anchor point** that defines the slot where the slave is available to receive communication
  - it responds in the immediately after slots

Sniff mode duty cycle

#card



Considering a device operating in sniff mode in a Bluetooth piconet. What is the duty cycle the device, assuming that it may both receive and send to the master at each anchor point?

- Note: consider only the duty cycle of the radio, disregard any other activity and assume the radio has to remain on for the entire duration of a slot.
- Hint: a slot in Bluetooth lasts 625 msec, although you can compute the duty cycle without this information
- ▶ ANSWER
  ^1623395938641

Hold mode
#card
acts like Sniff mode but only synchronous communication is possible

- if in two piconets, it can be used to make attend one of them
- to enter in low power mode

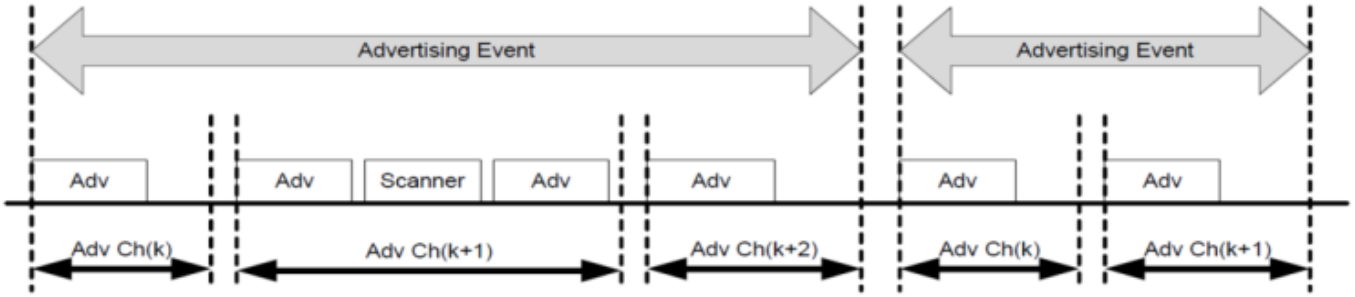
Park mode
#card

- gives away the address and communication is not possible anymore
- actively listen for sync messages from the master

LE

Advertising/scanning mechanism

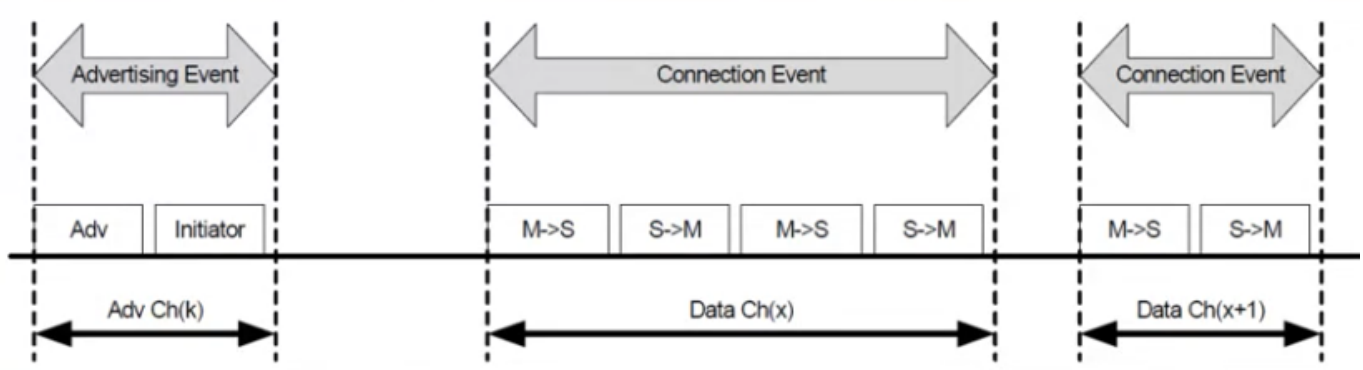
#card #exam



- devices uses just the last 3 channels (37, 38, 39)
- the advertiser publish an event all over the 3 channels
- a scanner can optionally respond on the same channel by asking more information, the advertiser then respond again

Piconet initialization

#card



- An **advertiser** issues a connectable advertising event
- The **initiator**, that is a **scanner** that is available to become the **master**, receives the event and starts the connection protocol by sending back the parameters (initiator rectangle, second from left)
- The **initiator** becomes the **master** of the piconet
- The **advertiser** becomes the **slave**

Roles

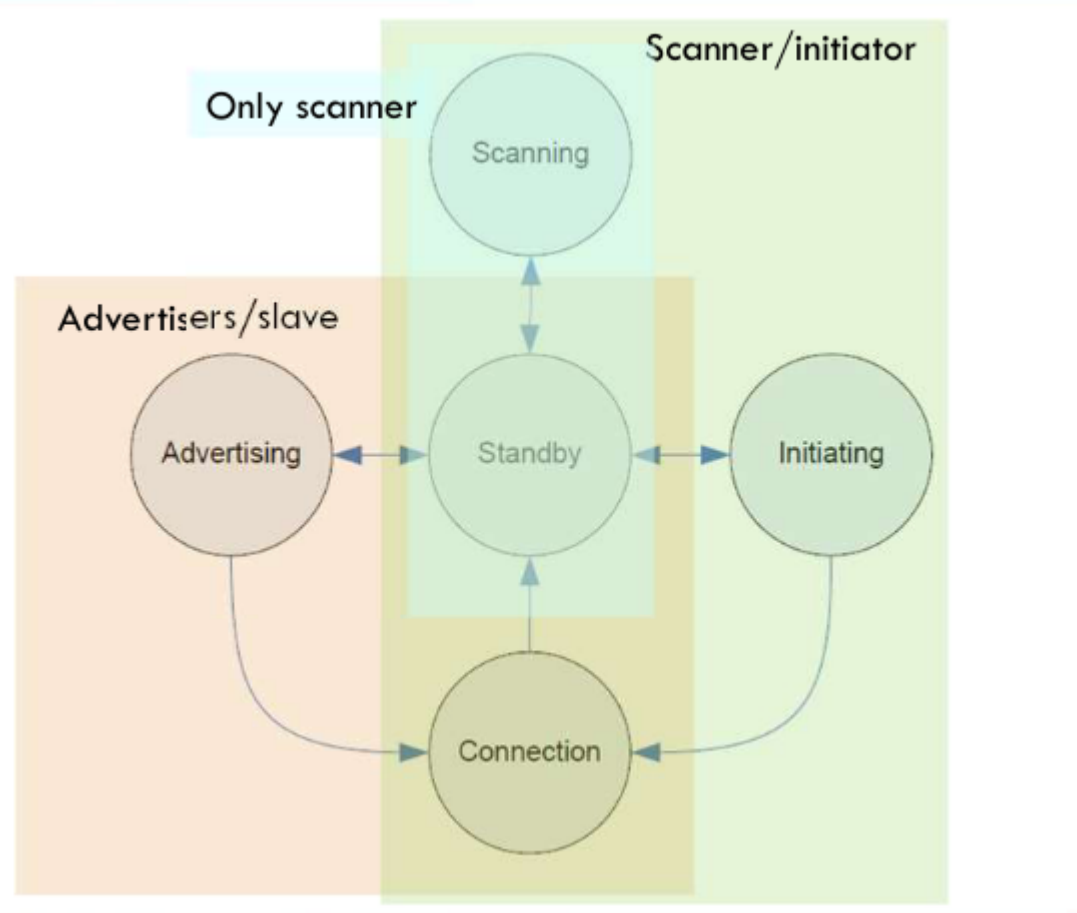
#card
A Bluetooth LE device can operate in four different device roles. Depending on the role, the devices behave differently.

- connection-based, acts like BR/EDR but only asynchronous communication
  - A **Peripheral device** is a **connectable advertiser** that can operate as a **slave** in a connection (e.g. a health thermometer or a heart rate sensor).
  - A **Central (initiator) device** scans for advertisers and initiate connections. It operates as a **master** in one or more connections (e.g. smartphones and computers).
- connection-less (one-directional communication), async communication with advertising events (*beacons*) over a physical link to *all* scanners
  - A **Broadcaster** is a **non-connectable advertiser** (e.g. a temperature sensor or an electronic tag).
  - An **Observer (scanner)** scans for advertisements, but cannot initiate connections (e.g. a remote display or a tracking system).

- a flag is used to indicate weather is connectable or not

States

#card



- based on the role that the device takes, it will take a subset of the following states
- **Observer scanner**: it will switch between *scanning* and *standby* in order to have a low duty cycle
- **Central initiator device**: as the observer but when it wants to create the network it will go the the *initiating* in order to see if there are available slaves, and then *connection*
- **Broadcaster**: switches between *advertising* and *standby*
- **Peripheral device (slave)**: as the broadcaster, but when the initiator responds with parameters, it will switch to *connection*↔ *standby*

Addresses

MAC

#card
MAC address: IEEE 48 bits:

- **BR/EDR**: pre-assigned (24 bits company name, 24 bits company ID)
- **LE**: Randomly generated (slave in a piconet) or pre-assigned (if advertiser, 24 bits company name, 24 bits company ID)

Piconet network

#card
Network addresses in the piconet

- **Active Member** address: 3 bits for active slaves, 0 for broadcast.
- **Parked Member** address: 8 bits

LMP/LLP (Data link - lower part)

#card
It's the **link manager layer** that controls and negotiates the connections with the slaves

- roles
  - piconet management (clock sync, switch master/slave)
  - setup Logical transport types
  - link configuration (create, QoS)
- protocols:
  - Link Management Protocol (LMP) for BR/EDR
  - Link Layer Protocol (LLP) for BLE

Host Controller Interface

#card
It's just a serial communication interface that allows the SW (host) to use the HW (controllers),

- **firmware** (controller), implement commands to access HW
- **driver** (host), manages the async events
- host controller transport layer, layer in between the previous two

L2CAP (Data link - upper part)

#card
Logical link control and adaptation protocol

- services
  - **protocol multiplex/demultiplex**, when a frame arrives from HW it needs to decide to which application protocol it must be delivered
  - **segmentation/reassembly** for data that is too long for a frame

    - Manages QoS and group communication
- no transmission of audio, no retransmission → no reliability

RFCOMM (Application core)

#card
Protocol that emulates serial line

Audio (Application core)

#card
No ACK because audio streams are in real time, so time would be wasted but correction with bits can be made

TCS (Application)

#card
Protocol to initiate calls

AT (Application)

#card
Protocol to control modem and phones

OBEX (Application)

#card
Object exchange protocol

ATT/GATT/GAP (Application core)

#card
Acts like the cluster library in ZigBee)

- **Attribute Protocol (ATT)**, defines access methods to the services in order to expose the state by showing attributes of the server to the client
    - attribute: identified with UUID or local name, permissions (read/write)
    - operations: get/set, push/pull (for sensors to obtain a simple **publish/subscribe**), broadcast
    - optional for BR/EDR
- **Generic Access Profile (GAP)** defines a list of protocols that any device should implement in order to support for a specific configuration **(what)**
    - For example: BR/EDR should have audio, baseband, LMP, L2CAP, SDP,

- **Generic Attribute Profile (GATT)** defines a profile for a service that is the list of attributes that provides
  - **service**: a collection of data and associated behaviors to accomplish a function of the device
    - is accepts ATT requests
    - e.g. computer (client) - temperature sensor (server)
  - optional for BR/EDR
- are supported by the device, because Bluetooth devices have same host, but the controllers are just a subset of all the possible one that can be implemented

SDP (Application core)

#card
Service discovery by asking to the server the available services and the attributes

- based on a client/server model by searching with a service UUID or get the whole list of services
- simpler than ZigBee, because it's a start network

SMP

Security management protocol uses the L2CAP channel for the

- pairing of devices (key generation, distribution)

Security

Goals

#card
Protections against

- passive eavesdropping;
- man-in-the-middle attacks;

Access model

#card
Pair a device with a piconet if they not share a key for mutual authentication (no crypto keys). Different types depending on the device's capabilities:

- **just works**
  - security is given intrinsically by the low range of devices (e.g. earbuds)

    - usually there is no display
- **numeric comparison**
    - display is needed to show a 6 digit number and let the user check weather it's the same on both devices
    - here the alternative channel is the human
- **out of band**
    - other channels (no bluetooth) to transfer the cryptographic codes
- **passkey entry**
    - one device with input and one with output capability. e.g. computer and bluetooth keyboard that needs to be paired

Secure pairing BR/EDR

1. Cryptographic keys

For the link-level security the following elements are needed:

- **BD_ADDR**, 48 bit
    - this can be easily eavesdropped with inquiry
- **Private key authentication (link key)**, 128 bit
    - obtained during initialization
    - semi-permanent or temporary (only one session of connection between master/slave)
- **Private key encryption**, 8-128 bit
    - the one used to encrypt
    - renewed at every session obtained from the authentication
- **Random n**, 128 bit

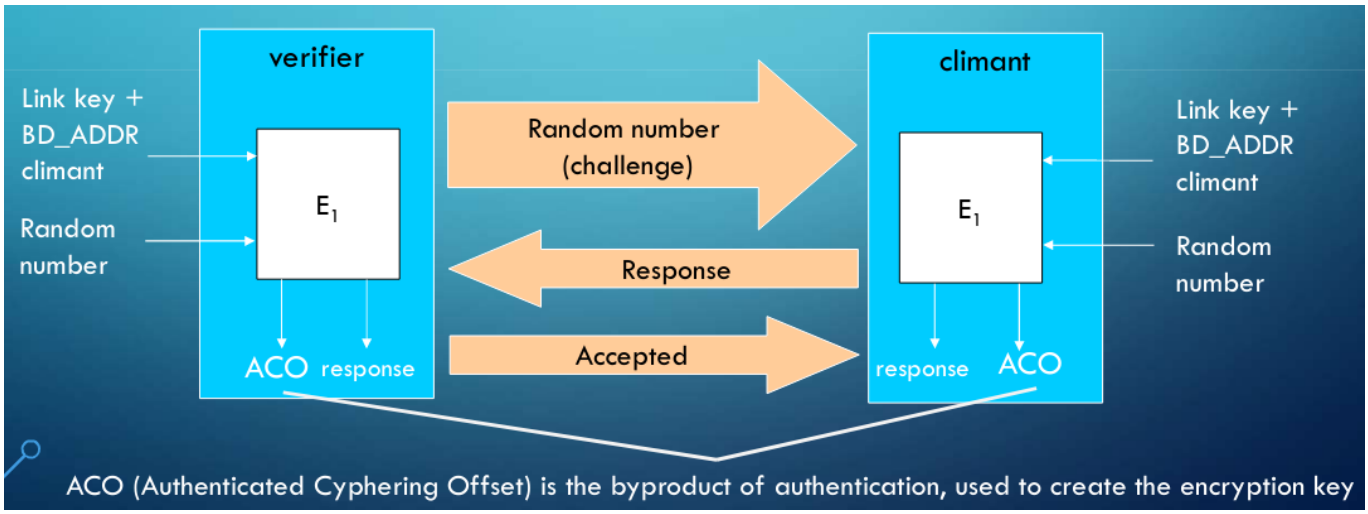
2. Authentication

Implemented at LMP layer to authenticate two devices when the slave wanna-be joins the network

- **mutual authentication**, first one direction and then the other one
- **mechanisms**
    - common link key
    - not have a common link key and they need to build it with a PIN

Common link key

#card #exam



Also called challenge/response mechanism

1. There are the verifier v and the climant c
2. c makes the request to v to be authenticated
3. v generates random number (**challenge**)
4. both of them know the Link key and the BD_ADDR of c
5. c receives the challenge and computes the ACO (Athenticated Cyphering Offset, namely the encryption key)
6. c sends the **response** to v
   v verifies and respond
7. if they match, now they share the same encryption key

3. Encryption

#card

Semi-permanent key	
Broadcast communications	Unicast communications
Not encrypted	Not encrypted
Not encrypted	Encrypted with the semi-permanent key

Temporary key, K_{master}	
Broadcast communications	Unicast communications
Not encrypted	Not encrypted
Not encrypted	Encrypted with the K_{master} key
Encrypted with the K_{master} key	Encrypted with the K_{master} key

Secure pairing BLE

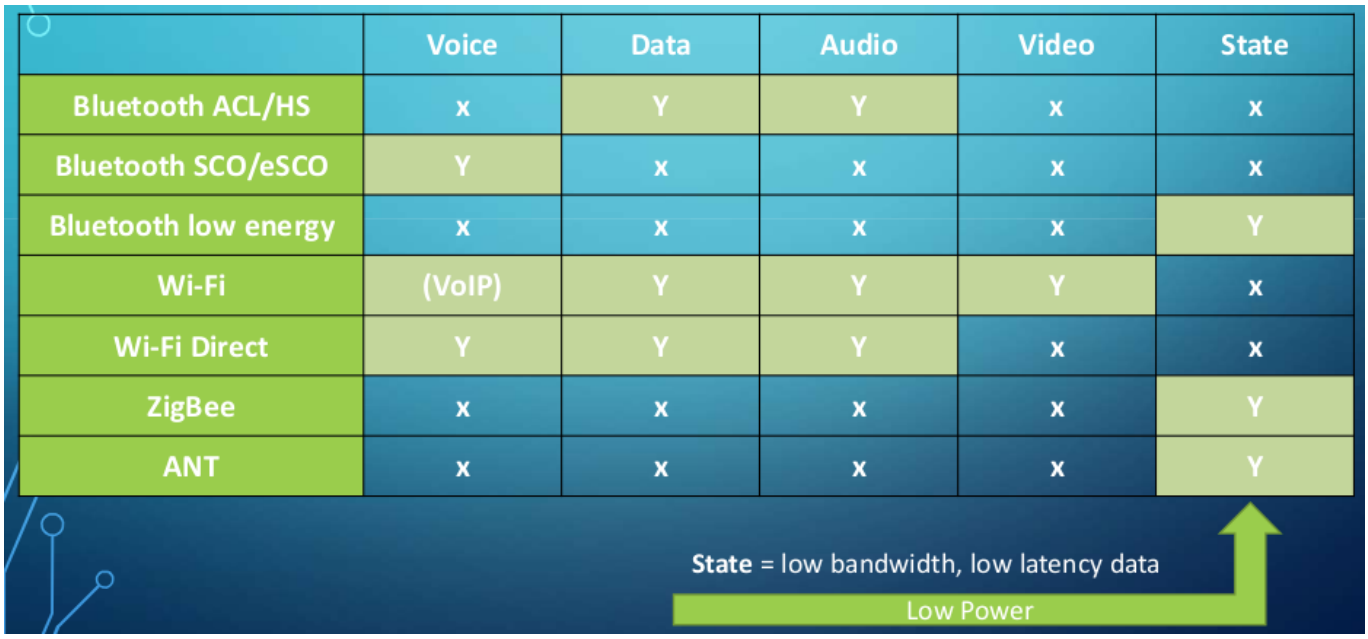
Requirements are totally different than Secure pairing BR EDR, because computational power is less

Privacy

To reduce the possibility of being tracked as a BLE device, the MAC can be changed frequently

- in order to being recognized by the "right" devices, the binding is done with private addresses

Comparisons



- bluetooth SCO/eSCO is the synchronized one used by BR/EDR

ZigBee

Compared to ZigBee

- Bluetooth is widely available in different kind of devices, ZigBee is constrained over IoT devices
- Beginning of 2000 there was the need in the E-health sector and IEEE 802.15.4 was not even there
- Only start topology
- Technically multi-hop is doable, but not known use-cases
- A device can switch between master/slave role