# COMP6733
# IoT Design Studio

## Week 2: Low Power Communications

# This lecture

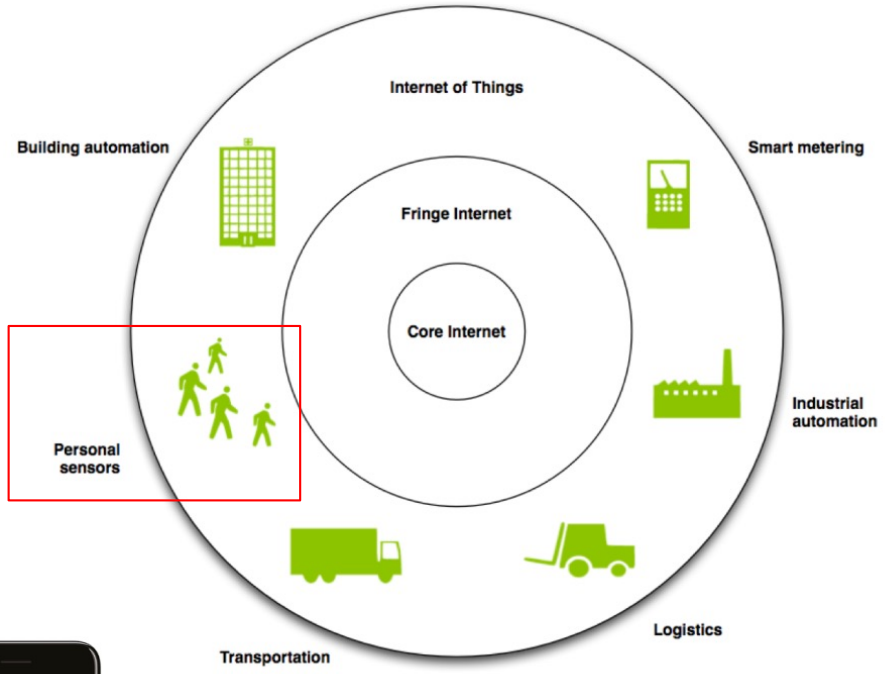
- Labs start this week
- Projects:
  - Topics
  - Groups

# This lecture

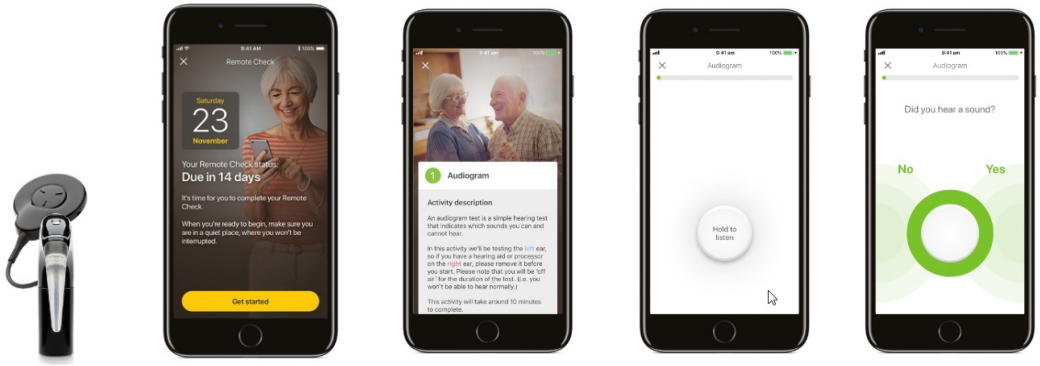
- BLE
- IEEE 802.15.4 (Zigbee)
- Link Layer and collection tree protocol
- 6LoWPAN and RPL

# BLE Example Applicationss

- Access to powerful sensor data and control:
  - https://good-design.org/projects/remote-hearing-health-check/
  - Audiometry




Source: "Embedded Web Services", Sensinode



# Intro: Bluetooth Classic

- The “conventional” Bluetooth
  - Also called Bluetooth BR (Basic Rate)/EDR (Enhanced Data Rate)
- 2.4GHz
- Range: 1m - 100m (10m typical)
- Connection-oriented: audio, file transfer, networking
- Reasonably fast data rate: 2.1 Mbps
- Power consumption:
  - High but still < Wifi < 3G

# How much energy does traditional Bluetooth use?

- Traditional Bluetooth is *connection oriented*. When a device is connected, a link is maintained, even if there is no data flowing.
- Sniff modes allow devices to sleep, reducing power consumption to extend battery life to a few months
- Peak transmit current is typically around 25mA
- Even though it has been independently shown to be lower power than other radio standards, it is still not low enough power for coin cells and energy harvesting applications

# What is Bluetooth Low Energy?

- Bluetooth low energy is a NEW, open, short range radio technology
  - Blank sheet of paper design
  - Different to Bluetooth classic (BR/EDR)
  - Optimized for ultra low power
  - Enable coin cell battery use cases
    - < 20mA peak current
    - < 5 uA average current


- Target applications
  - Wireless battery-powered sensors, e.g. heart rate, thermometer, fitness
  - Location tracking and information serving, e.g. iBeacons
  - Internet of Things
- Requirements for target applications
  - Low-power
  - Low-cost
  - Low bandwidth: ~100 kbps
  - Low latency: Connectionless (fast setup and teardown of connection in ~10ms)

# Bluetooth Classic vs Bluetooth Low Energy

- Bluetooth Classic:
  - Historical competitor to WiFi
  - Power hungry
  - Audio
- Bluetooth Low Energy, starting from 4.0:
  - Ultra low power consumption
  - Restricted packet length
  - Less channels, fatter signals (higher modulation index)
  - Majority of time is sleeping
  - Very quick in creating connections
  - No audio officially in the spec, but has been done in hearing aid industry.


# Basic Concepts of Bluetooth 4.0

- Everything is optimized for lowest power consumption
  - Radio chip off most of the time
  - Small packets
    - MTU: 20 bytes/packet for application
    - Less time transmitting -> less heat -> no need compensatory circuits -> save more power
  - Short packets reduce TX peak current
  - Short packets reduce RX time
  - Less RF channels to improve discovery and connection time
  - Simple state machine
  - Single protocol
  - Etc.

# Bluetooth low energy factsheet

| | |
|---|---|
| Range: | ~ 150 meters open field |
| Output Power: | ~ 10 mW (10dBm) |
| Max Current: | ~ 15 mA |
| Latency: | 3 ms |
| Topology: | Star |
| Connections: | > 2 billion |
| Radio Frequency: | ISM 2.4 GHz band |
| Robustness: | Adaptive Frequency Hopping, 24 bit CRC |
| Security: | 128bit AES CCM |
| Sleep current: | ~ 1µA |
| Modes: | Broadcast, Connection, Event Data Models, Reads, Writes |

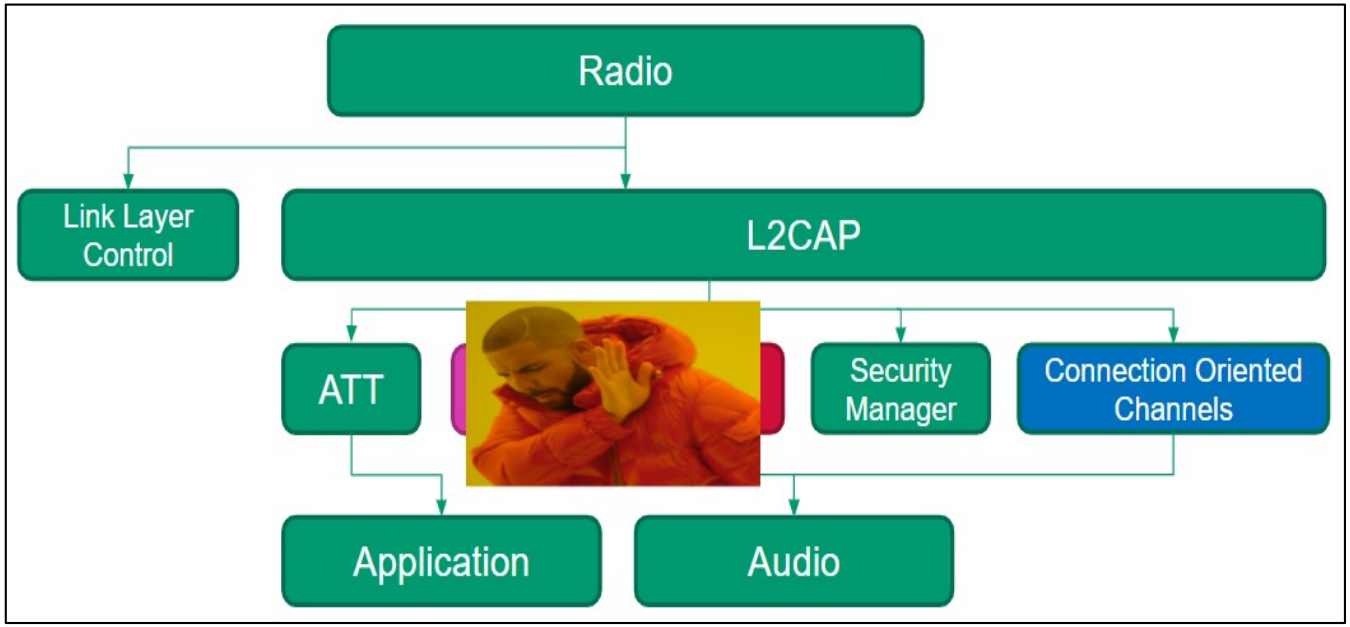
- Data Throughput
  - For Bluetooth low energy, data throughput is not a meaningful parameter. It does not support streaming.
  - It has a data rate of 1Mbps, but is not optimized for file transfer.
  - It is designed for sending small chunks of data (exposing state)

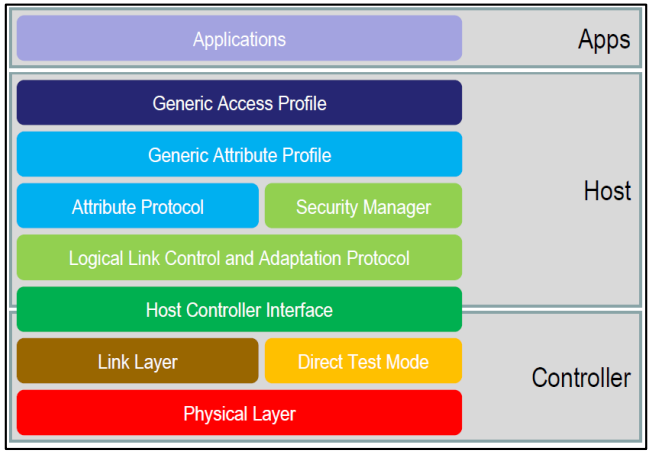
# Designed for exposing state


- It's good at small, discrete data transfers.
- Data can triggered by local events.
- Data can be read at any time by a client.
- Interface model is very simple

# Protocol Stack

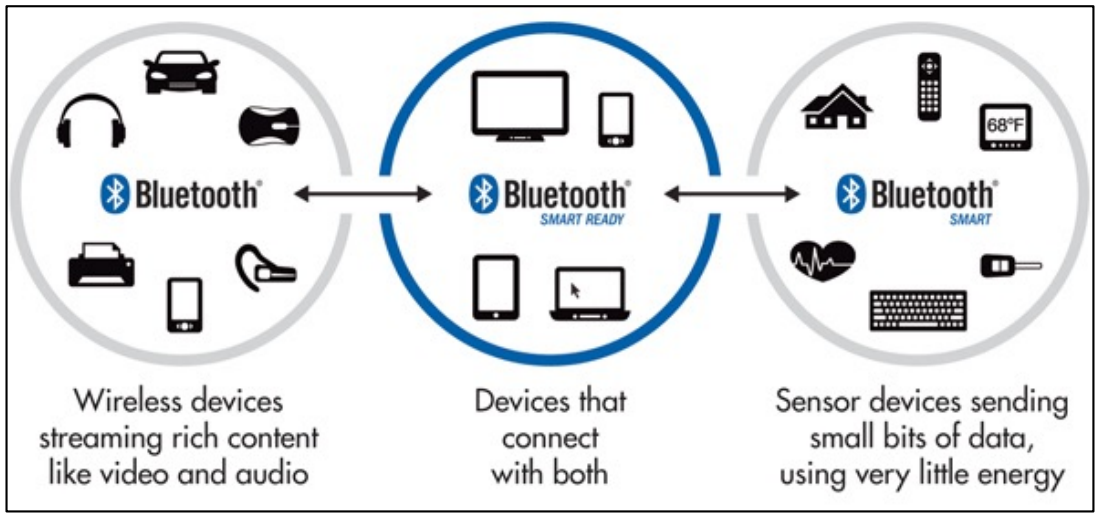
- Big companies have their own Bluetooth stack
  - Talk to each other referring to spec, winking when talking about the same thing on our own secret implementation.
- Off the shelf solutions go through SDK (Soft Device)
  - Effort to protect their implementation.



# Device Modes and Applications

- Dual Mode
  - Bluetooth BR/EDR and LE
  - Used anywhere that BR/EDR
  - is used today
- Single Mode
  - Implements only Bluetooth low energy
  - Will be used in new devices/applications



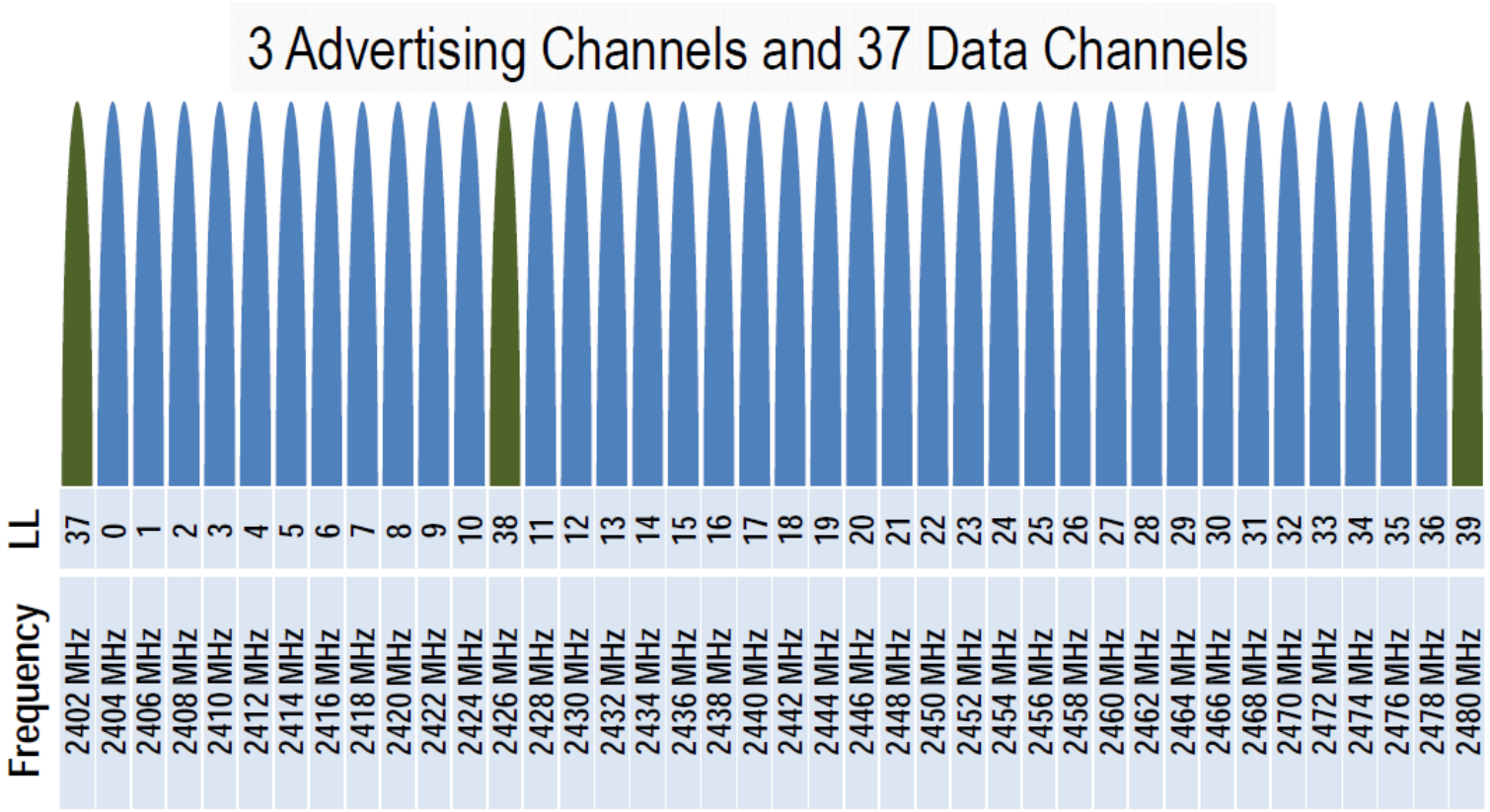
# Slave has two main states

- Advertising:
  - Beacons: no intention to connect to a master device.
    - Smart bathroom scale might do this.
    - BLE 5 introduced Extended Advertising PDU (protocol data units) fitting 254 bytes of data.
  - Getting into a connected state:
    - May allow new pairing/bonding.
    - Allow reconnections to previously bonded device only.
    - Advertise random data only a previously connected data will recognise.

- Connected state:
  - Persistent connection:
    - Session key for security.
    - Aligned in terms of **timing**, **frequency**, and **access address**.
    - Powerful control over packet lengths and timing.
  - Standardised BLE profiles:
    - Heart rate monitors may all implement same BLE profile.
    - Define specification between Master and Slave devices.
- Bonded state: encryption enabled

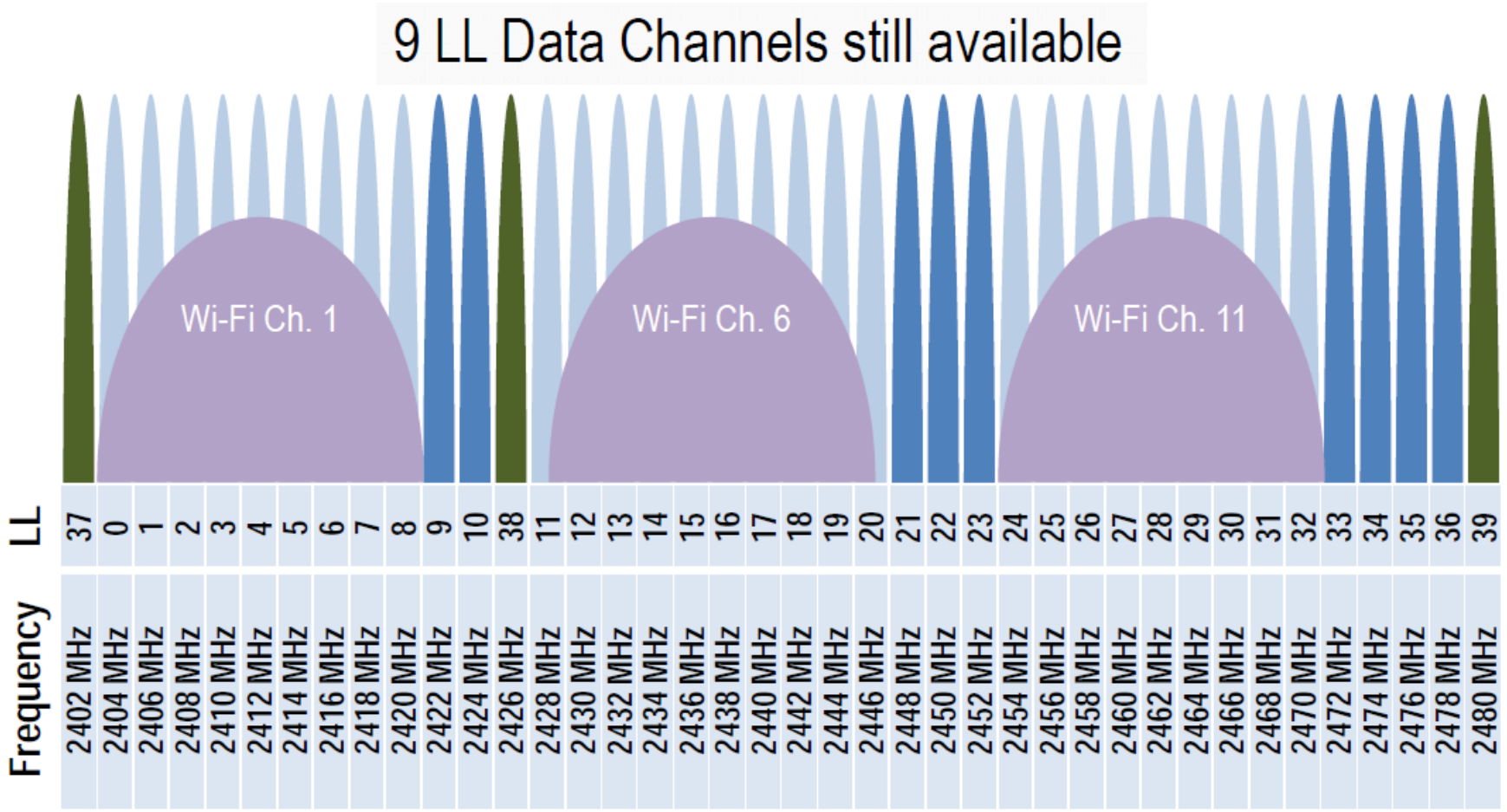
# Physical Layer

- 2.4 GHz ISM band
- 1Mbps
- 40 Channels on 2 MHz spacing



# Physical Channels

- Advertising channels avoid 802.11

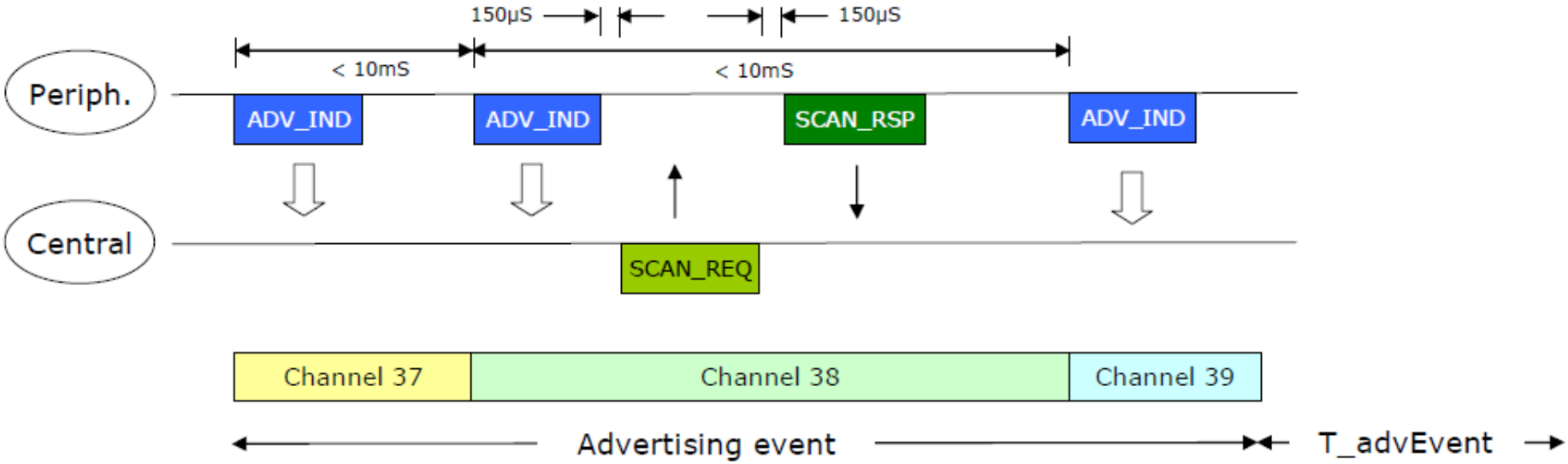


# Link Layer

- Low Complexity
  - 1 packet format
  - 2 PDU types –depending on Advertising/Data Channel
  - 7 Advertising PDU Types
  - 7 Link Layer Control Procedures

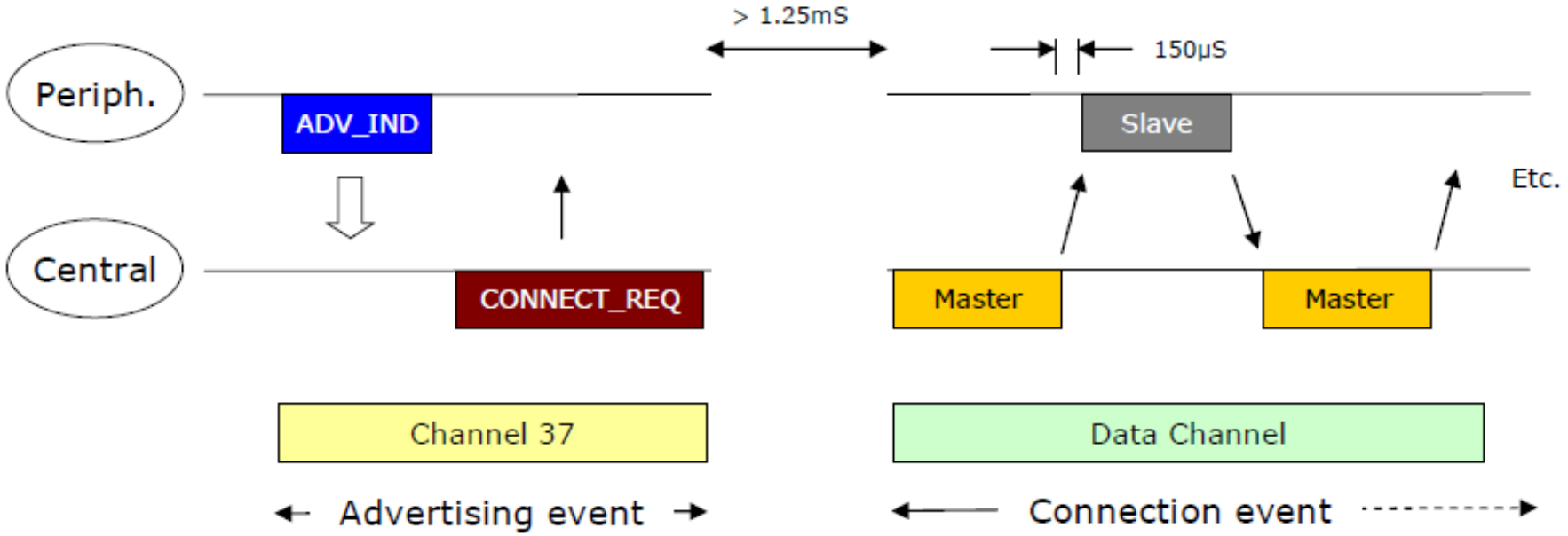
- Useful Features
  - Adaptive Frequency Hopping
  - Low Power Acknowledgement
  - Very fast connections

# Advertising




- Devices can advertise for a variety of reasons:
  - To broadcast promiscuously
  - To transmit signed data to a previously bonded device
  - To advertise their presence to a device wanting to connect
  - To reconnect asynchronously due to a local event

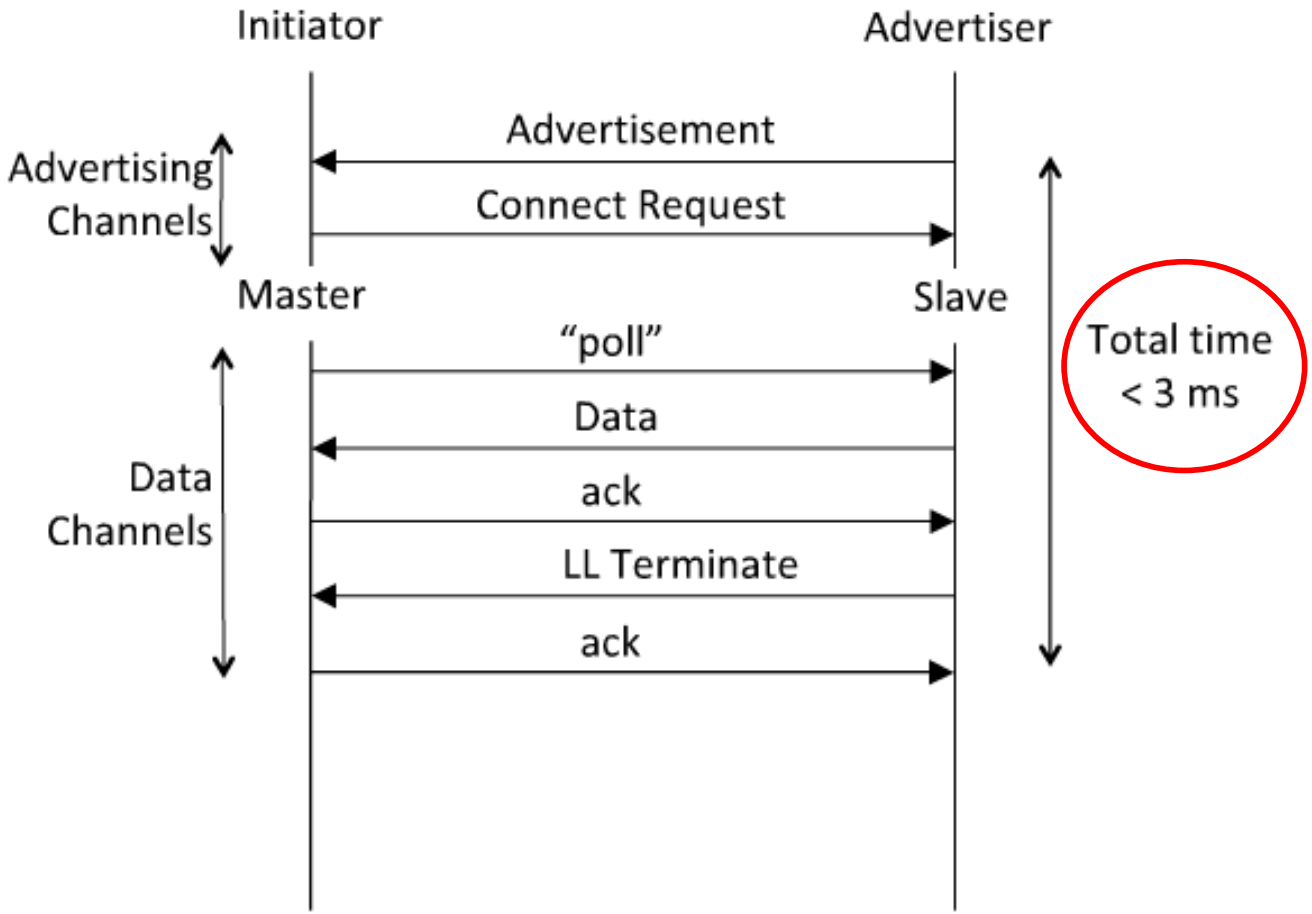
# Data transactions




- Once a connection is made:
  - Master informs slave of hopping sequence and when to wake
  - All subsequent transactions are performed in the 37 data channels
  - Transactions can be encrypted
  - Both devices can go into deep sleep between transactions

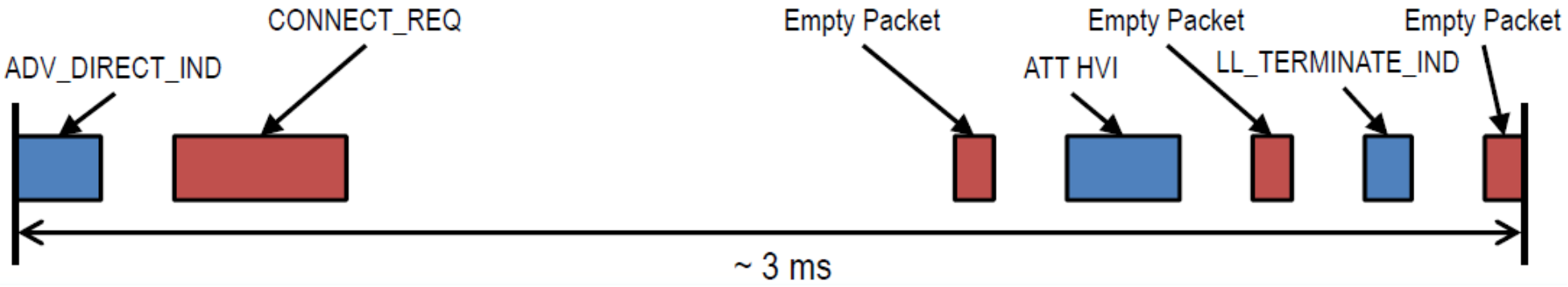
# Link Layer Connection

- Very low latency connection



# Time From Disconnected to Data ~ 3ms

| Time (us) | Master Tx | Radio Active (us) | Slave Tx |
|---|---|---|---|
| 0 | | 176 | ADV_DIRECT_IND |
| 326 | CONNECT_REQ | 352 | |
| 1928 | Empty Packet | 80 | |
| 2158 | | 144 | Attribute Protocol<br>Handle Value Indication |
| 2452 | Empty Packet (Acknowledgement) | 80 | |
| 2682 | | 96 | LL_TERMINATE_IND |
| 2928 | Empty Packet (Acknowledgement) | 80 | |



# Link Layer: Staying Connected

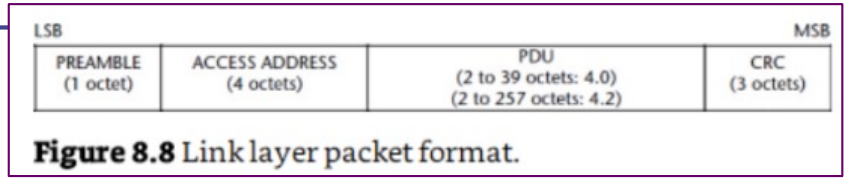
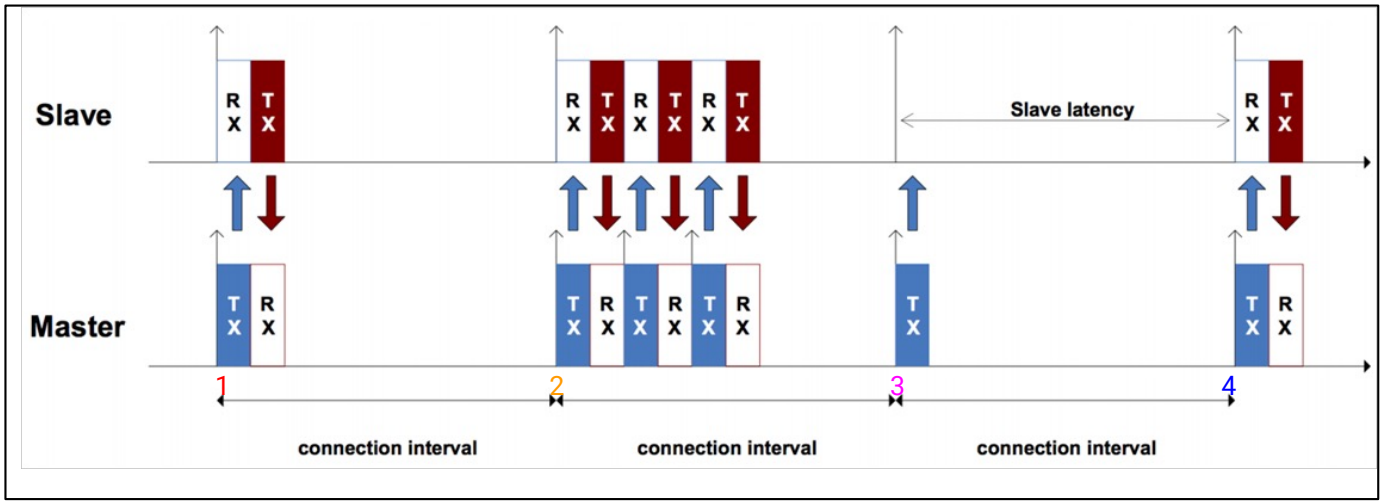



**Figure 8.8** Link layer packet format.

1. Connection interval: Master first, Slave second.
2. More Data (MD) allows more transactions.
3. Slave Latency: Slave can choose not to wake up. Saves power.
   - Example: Teacher must come to class, students may skip some classes.
4. Connection interval again.



# Link Layer: Staying in Sync

- When no data to send:
  - Empty PDU (Protocol Data Unit) will bear only link layer (2 byte) header.
- Packets are acknowledged before the next one can be transferred:
  - Sequence number (SN), a bit
  - Next Expected Sequence Number (NESN), a bit
  - Slave receives a PDU from Master.
  - Master's SN matches Slaves NESN.
  - Slave toggles NESN bit in next packet sent to Master.
  - Master sees Slave ready for next PDU.
  - Master toggles SN and sends new PDU in next connection event.
- Connection Request contains connection timeout:
  - When either side disappears from going out of range.
  - When SN and NESN isn't honored.

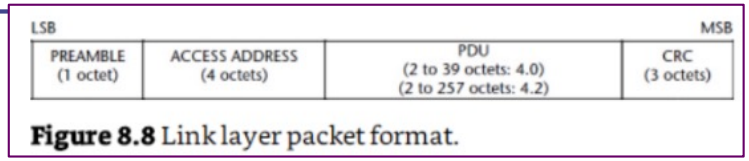


**Figure 8.8** Link layer packet format.

Good:

| | SN | NESN |
|---|---|---|
| Master | 0 | 0 |
| Slave, received (SN ==NESN) | 0 | 1 - increment |
| Master, received (SN!=NESN) | 1 - Toggles | 1 |
| Slave, received (SN==NESN) | 1 | 0 - Toggles |

Bads:

| | SN | NESN |
|---|---|---|
| Master | 0 | 0 |
| Slave, not received / toggled | 0 | 0 – No toggle |
| Master | 0 | 1 |
| Slave | 1 | 0 |
| Master | 0 | 0 |

# Link Layer: Control

- Feature Request:
  - Length Extension (post BLE 4.0).
  - PHY changes (post BLE 5.0).
  - Security, pairing
- Many settings have countdowns:
  - Starting encryption. Requires help from Security Manager .
  - Changing connection interval
  - Changing PHY (Physical Layer)
  - Whitelisting channels via channel map.
    - Master device (phone) has to figure this out

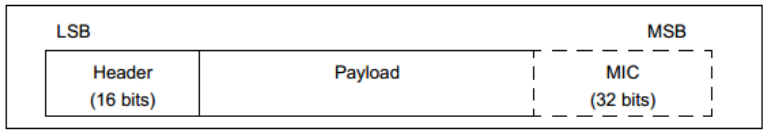



*Figure 2.12: Data Channel PDU*

| Header | | | | | |
|---|---|---|---|---|---|
| LLID (2 bits) | NESN (1 bit) | SN (1 bit) | MD (1 bit) | RFU (3 bits) | Length (8 bits) |

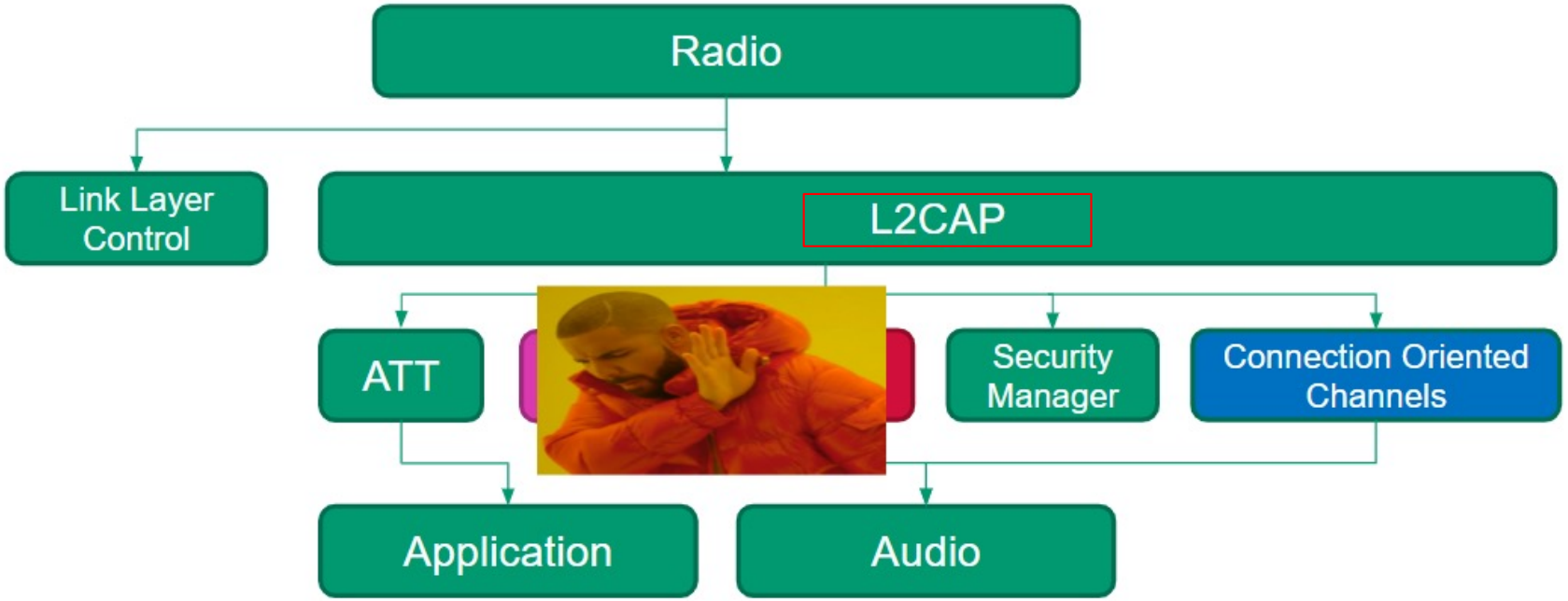
*Figure 2.13: Data channel PDU header*

| Field name | Description |
|---|---|
| **LLID** | The LLID indicates whether the packet is an LL Data PDU or an LL Control PDU.<br>00b = Reserved<br>01b = LL Data PDU: Continuation fragment of an L2CAP message, or an Empty PDU.<br>10b = LL Data PDU: Start of an L2CAP message or a complete L2CAP message with no fragmentation.<br>11b = LL Control PDU |
| **NESN** | Next Expected Sequence Number |
| **SN** | Sequence Number |
| **MD** | More Data |
| **Length** | The Length field indicates the size, in octets, of the Payload and MIC, if included. |

*Table 2.3: Data channel PDU Header field*

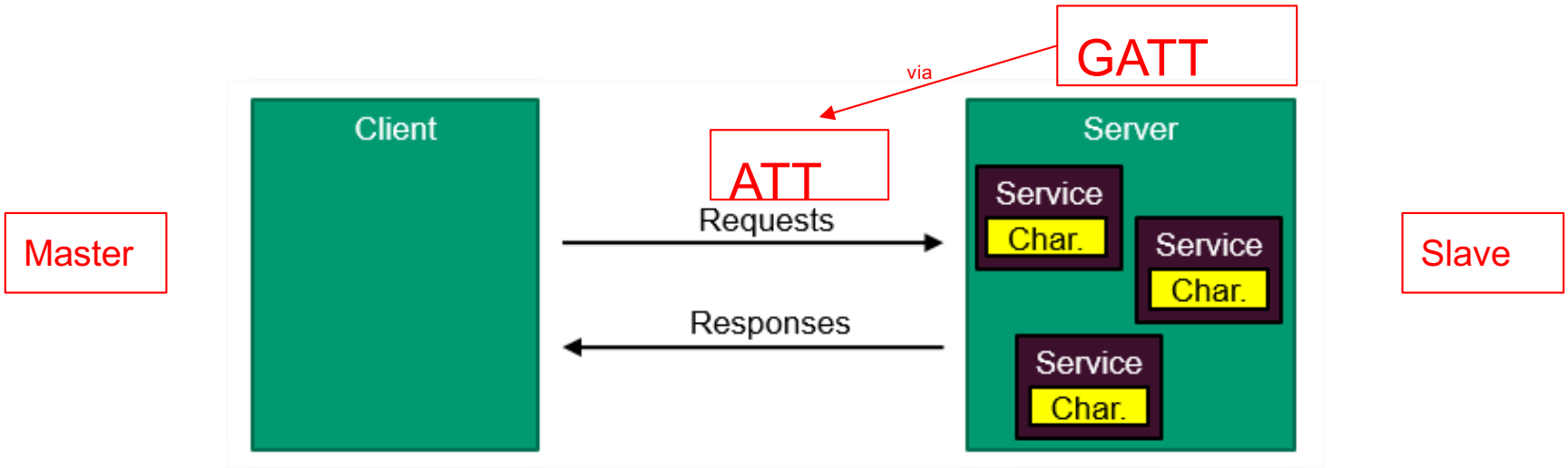
# L2CAP

- Logical Link Control and Adaptation Protocol.
  - Multiplexor
  - Deal with packet splitting and recombining.



# Generic Attribute Profile (GATT)

- Organising data:
  - Data is exposed in “Characteristic”
  - These are bunched up in “Services”:
    - Generic Attribute Profile Service: GattServiceChanged
    - Generic Access Profile Service: Name
    - Device Information Service: Manufacturer name



# Services and Characteristics (GATT)

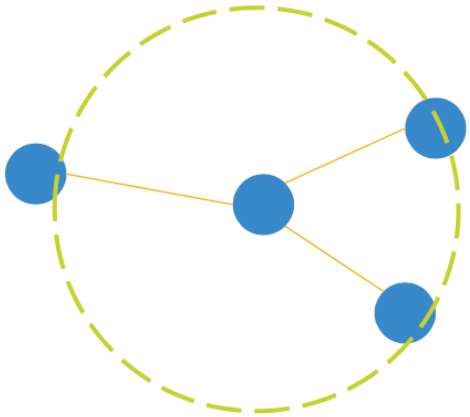
- Services and Characteristics are recognised by UUIDs.
- Example: Master looks for a Heart Rate **Service** by reading handle 0x000E and finding the UUID 0x180D:
  - Within this service, read the **Characteristic** entry and finding the UUID (0x2A37):
    - Recognise the UUID and know it is Heart Rate.
    - On Handle 0x0010
    - **Notifiable** only. Other Characteristics may allow **Read** or **Write**.
  - Configuration of the **Notification** is then on the next handle (0x0011):
    - Enable Notification by **Writing** to CCCD: Client Characteristic Configuration Descriptor).
  - When we receive a notification on 0x0011, we know it's Heart Rate. No need for UUIDs from this point.

| Heart Rate Profile | Handle | Type of attribute (UUID) | Attribute permission | Attribute value |
|---|---|---|---|---|
| Service Declaration | 0x000E | Service declaration<br>Standard UUIDservice 0x2800 | Read Only, No Authentication, No Authorization | Heart Rate Service 0x180D |
| Characteristic Declaration | 0x000F | Characteristic declaration<br>Standard UUIDcharacteristic 0x2803 | Read Only, No Authentication, No Authorization | Properties (Notify) Value Handle (0x0010) UUID for Heart Rate Measurement characteristic (0x2A37) |
| Characteristic Value Declaration | 0x0010 | Heart Rate Measurement Characteristic<br>UUID found in the Characteristic declaration value<br>0x2A37 | Higher layer profile or implementation specific. | Beats Per Minute E.g "167" |
| Descriptor Declaration | 0x0011 | Client Characteristic Configuration Descriptor (CCCD)<br>Standard UUIDservice 0x2800 | Readable with no authentication or authorization. Writable with authentication and authorization defined by a higher layer specification or is implementation specific. | Notification enabled 0x000X |

# Bluetooth Mesh

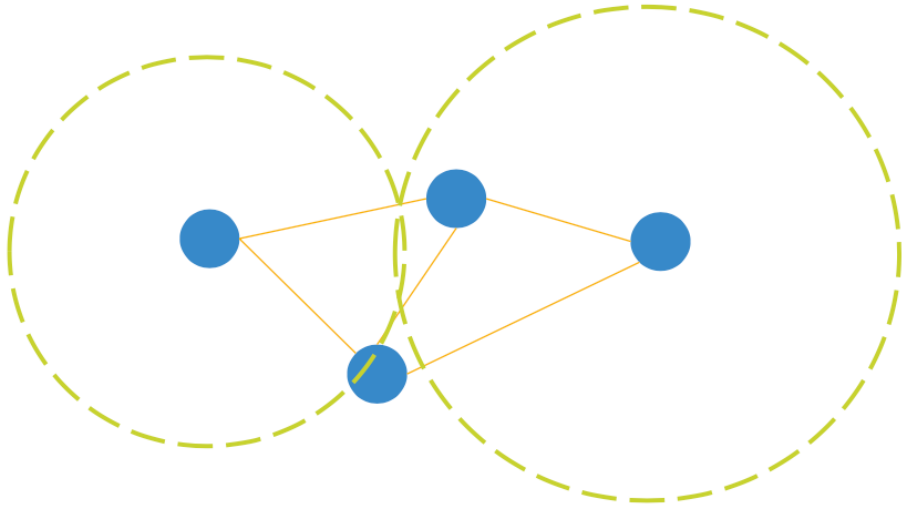
## Bluetooth LE

- **Two states** of a device: advertising and connected
- **Topology**: point-to-point or at best star
- **Range**: line of sight: protocol allows a datagram to be exchanged by two neighbor devices



## Bluetooth LE Mesh

- **A single state** of a device: advertising
- **Topology**: flooded mesh
- **Range**: configurable: protocol allows datagram to be relayed as many times as desired



# How low can the energy get?

- From the previous slide, calculate energy per transaction
  - Assume an upper bound of 3ms per minimal transaction
  - Estimated TX power is 15mW (mostly TX power amp for 65nm chips)
  - For 1.5v battery, this is 10mA. 0.015W * 0.003 sec = 45 micro Joule
- How long could a sensor last on a battery?
  - An example battery: Lenmar WC357, 1.55v, 180mAh, $2-5
  - 180mAh/10mA = 18Hr = 64,800 seconds = 21.6M transactions
  - Suppose this sensor sends a report every minute = 1440/day
  - For just the BT LE transactions, this is 15,000 days, or > 40 years
  - This far exceeds the life of the battery and/or the product
- This means that battery will cost more than the electronics
  - This sensor could run on scavenged power, e.g. ambient light

# Bluetooth Low Energy: Summary

- Emphasis on low energy
- Particularly suited for the Internet of Things ecosystem
  - To connect every day things that we carry – watches, wearable tech, body sensors, fobs, home and office automation
- New standard designed for a new decade of connected devices

- Details: https://www.bluetooth.org/en-us/specification/adopted-specifications