Wireless Communication & Bluetooth Low Energy

Project-based Learning Center | ETH Zurich
Michele Magno michele.magno@pbl.ee.ethz.ch
Tommso Polonelli tommaso.polonelli@pbl.ee.ethz.ch
Vlad Niculescu vladn@iis.ee.ethz.ch

Credits: STMicroelectronics.

Program and structure of the course

DATE	Topic	
04.03.2021	Introduction to the course • Microcontrollers in general and STM32 • Integrated Development Environment – STM32CubeIDE • LAB1: How to program an STM32	TP
11.03.2021	PWM and motors	TP
18.03.2021	Serial interfaces UART + SPI + I2C + IMU	TP
25.03.2021	EKF and AHRS	VN
01.04.2021	Bluetooth Low Energy	TP
15.04.2021	Control PID, attitude control	VN
22.04.2021	**Project presentation (Guest)** **Project selection and kick start**	TP/Guest
29.04.2021	Flight test (drone test room) **Exercise summary**	TP/VN
	Project	TP
	Project	TP
	Project	TP
	Project	TP
03/06/2021	**Final presentation**	MM/TP

Wireless Sensor Network Protocols

- WSN protocol strongly impacts system performance
- Choosing the wrong protocol may cause inefficiency and prevent the WSN to accomplish user need.

The protocol affect:

- Energy dissipation
- System cost
- Latency
- Security

IEEE

ZigBee™ Alliance

Bluetooth

sigfox

ISA100 Wireless COMPLIANT

WirelessHART

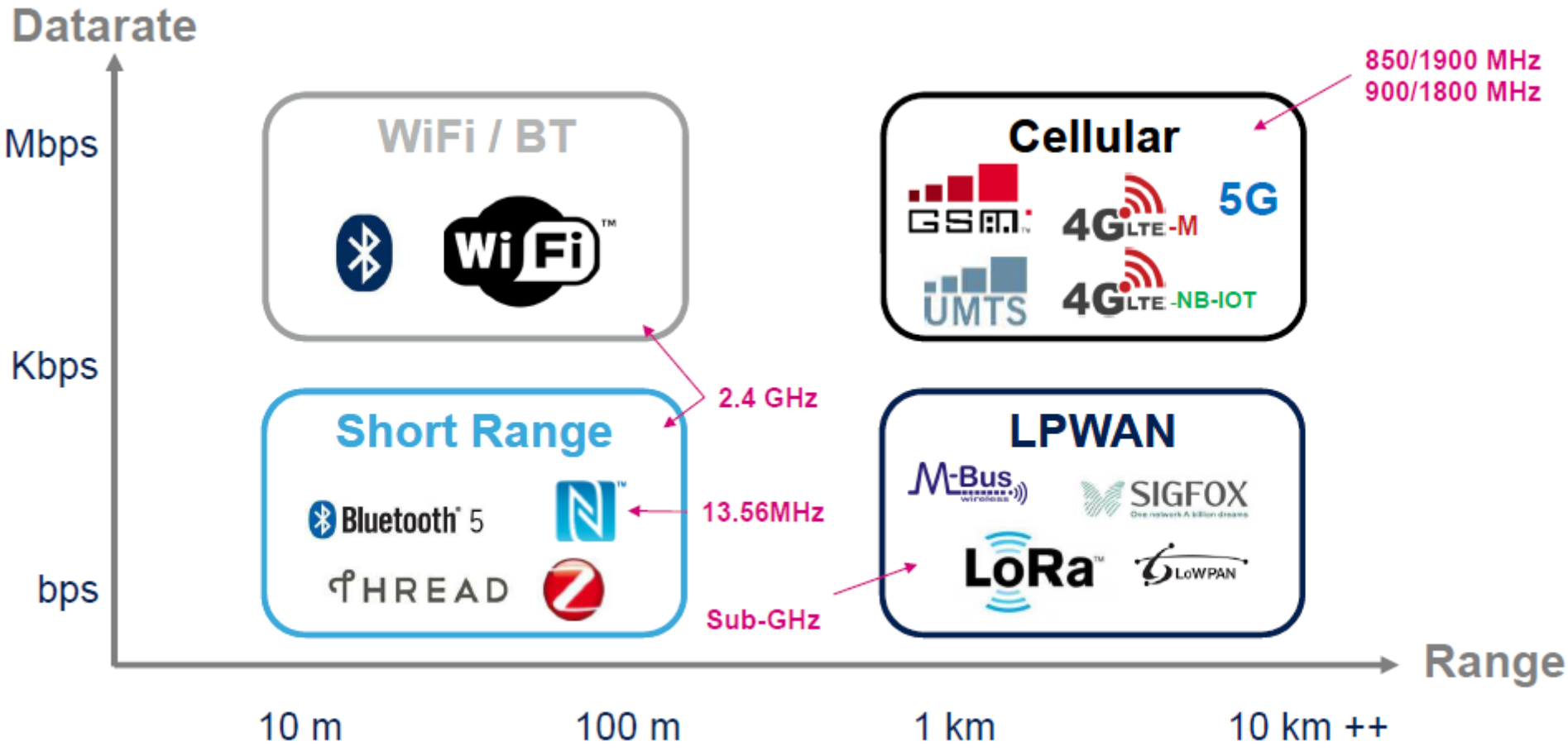
wibree

6LoWPAN
IPv6-based Low-power Wireless Personal Area Networks

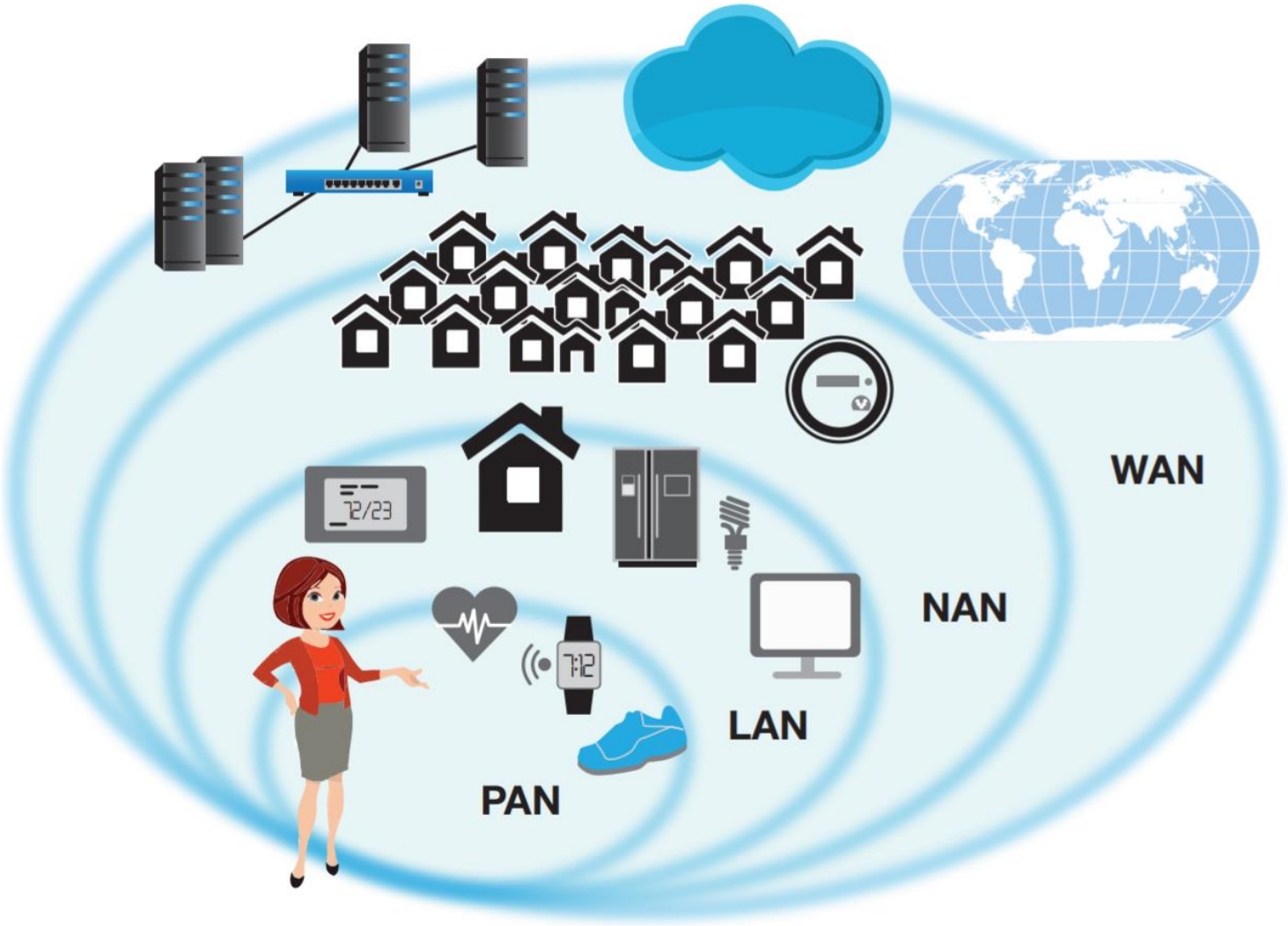
ANT
the power of less

LoRa™

Wireless Sensor Network Protocols



Networks Range

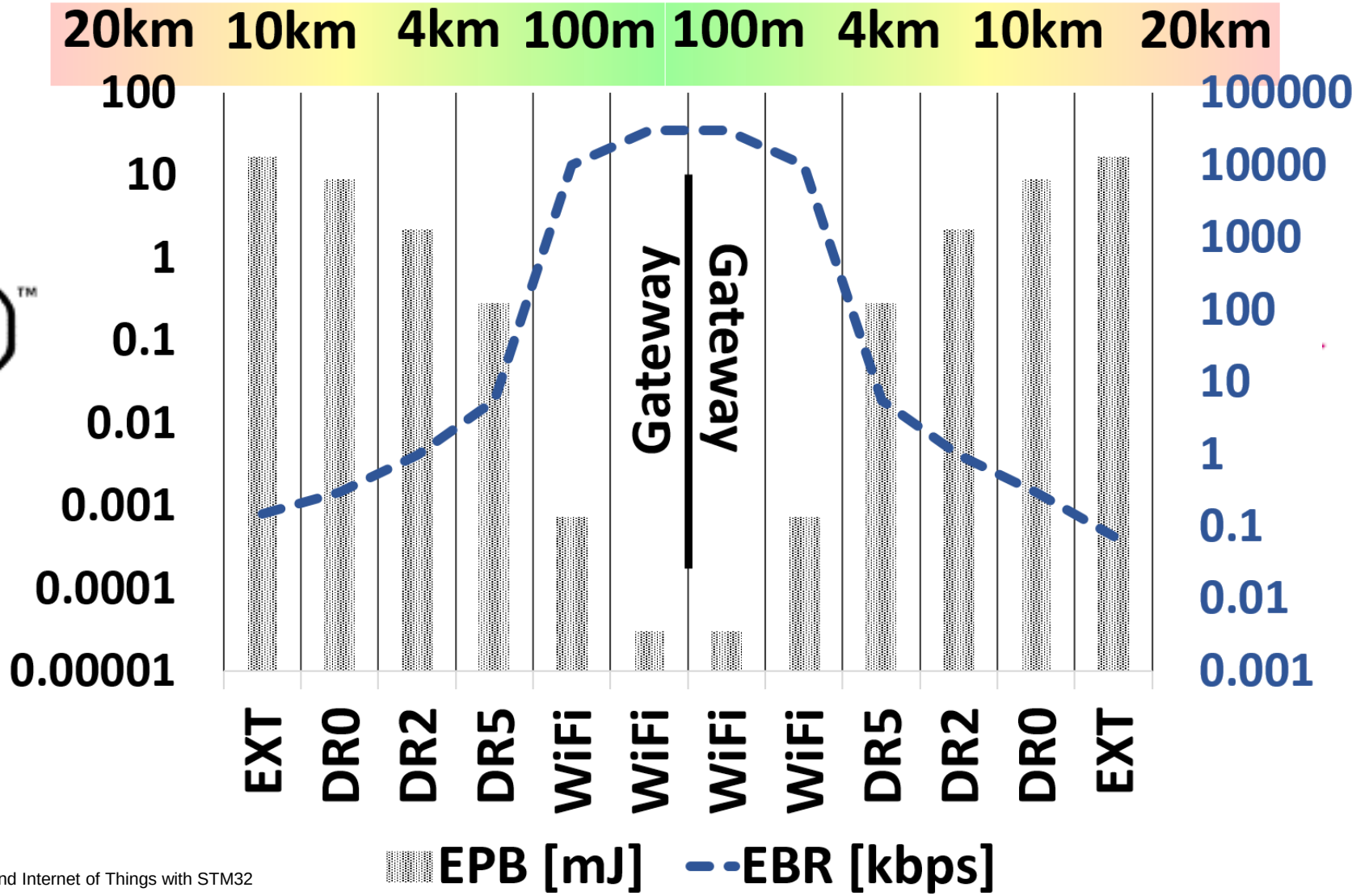


Network's range is typically categorized into:

- Personal area networks (PANs)
- Local area networks (LANs)
- Neighborhood area networks (NANs)
- Wide area networks (WANs).

Speed v.s. Energy v.s. Coverage

- EPB: Energy Per Bit
- EBR: Equivalent Bit Rate



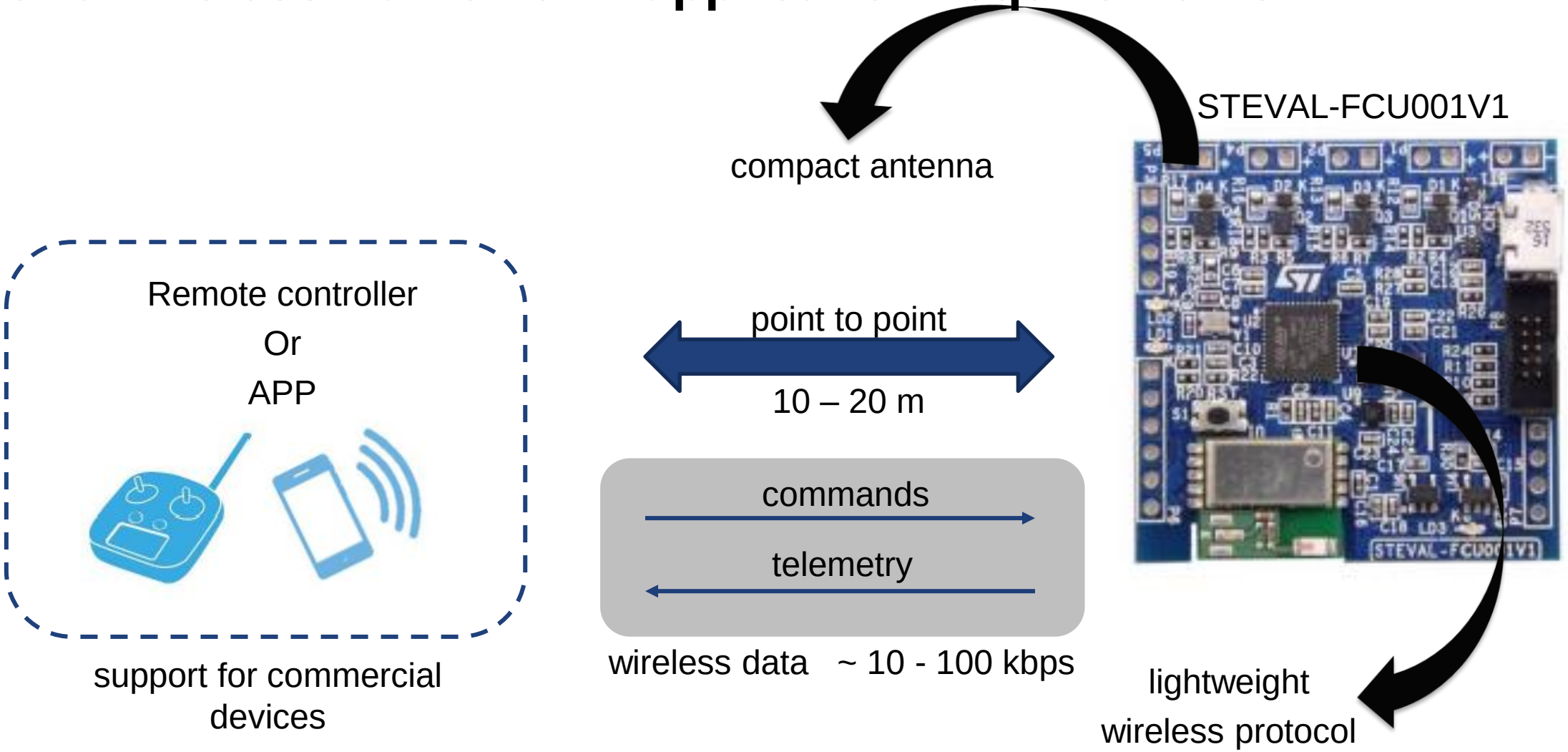
Key considerations influencing the choice of wireless protocols for a specific application

	Wi-Fi*	BLE/ Bluetooth 5	Thread	Sub-1 GHz: TI 15.4	Sub-1GHz: Sigfox	Zigbee
Data throughput	Up to 72Mbps	Up to 2Mbps	Up to 250kbps	Up to 200kbps	100bps	Up to 250kbps
Range**	100m	Up to 750m	100m via mesh	4km	25km	130m LOS
Power consumption	Up to 1 year on AA batteries	Up to years on a coin-cell battery	Up to years on a coin-cell battery	Up to years on a coin-cell battery for 1km range	Up to years on a coin-cell battery for limited range	Years on a coin-cell battery
Topology	Star	Point-to-poin/Mesh	Mesh & Star	Star	Star	Mesh & Star
IP at the device node	Yes	No	Yes	No	No	No
PC, mobile OS support	Yes	Yes	No	No	No	No
Infrastructure widely deployed	Yes, Access Points	Yes, smart phones	No	No	No	No

**Single stream 802.11n Wi-Fi MCUs may support lower throughput than peak physical capacity of the network.*

***LOS = Line Of Sight. For range, note that maximum data rates are often not available at the longest range.*

Drone wireless framework: application requirements



Drone wireless framework: application requirements

Remote controller

Or

APP

support for commercial devices

compact antenna

STEVAL-FCU001V1

point to point

10 – 20 m

commands

telemetry

wireless data ~ 10 - 100 kbps

lightweight wireless protocol

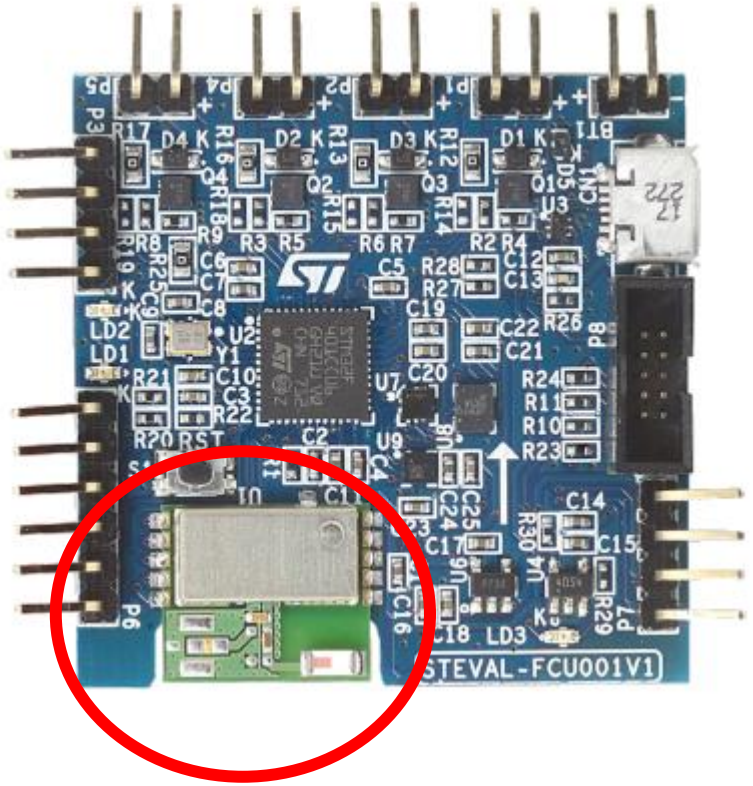
STEVAL-FCU001V1

Key Features

- Compact flight controller unit (FCU) evaluation board complete with sample firmware for a small or medium sized quadcopter
- On-board LiPo 1-cell battery charger
- Possibility to directly drive 4 DC brushed motors through the low voltage onboard MOSFET or alternatively use external ESC for DC brushless motor configuration

Main Components

- STM32F401 – 32-bit MCU with ARM® Cortex®
- LSM6DSL – iNEMO intertial module: 3D accelerometer and 3D gyroscope
- LIS2MDL – High performance 3D magnetometer
- LPS22HD – MEMS pressure sensor: 260-1260 hPa absolute digital output barometer
- **SPBTLE-RF – Very low power module for Bluetooth Smart v4.1**
- STL6N3LLH6 - N-channel 30 V, 6 A STripFET H6 Power MOSFET
- STC4054 - 800 mA standalone linear Li-Ion battery charger



Bluetooth Low Energy module

Bluetooth Low Energy Module: BLUENRG-M2

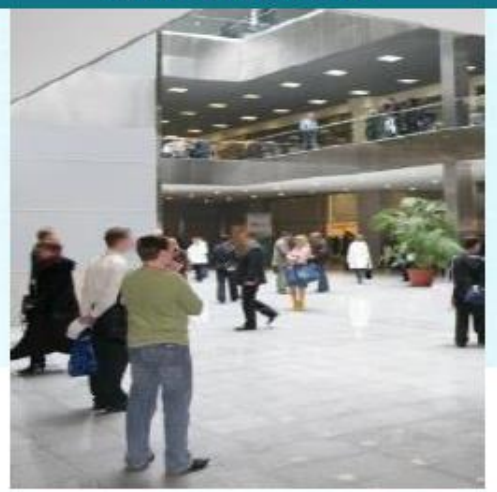
The entire Bluetooth Smart stack and protocol are embedded into the BLUENRG-M2 module

- Bluetooth V4.1 compliant (supports master and slave modes, multiple roles supported simultaneously
- Embedded Bluetooth low-energy protocol stack (GAP, GATT, SM, L2CAP, LL, RFPHY)

- Bluetooth radio performance:
  - Embedded ST BlueNRG-MS
  - Tx power: + 8 dBm
  - Host interface: SPI, IRQ, and RESET. On-field stack upgrading available **via SPI.**
  - On-board chip antenna

- IDEAL for short range communication (10-50m) and connection with smartphones

Wireless Connectivity is *EVERYWHERE*

Automotive	Motor drives	Building automation	Smart grid	Factory automation
				

Wearables	Audio	Home Electronics	Smart Peripherals
			

Bluetooth low energy vs other Bluetooth spec

Bluetooth® Classic	Bluetooth® Dual Mode	Bluetooth® Low Energy (BLE)
		
Power Required: AAA	**Power Required:** AAA	**Power Required:** Coin cell
• Optimized for audio applications • Higher throughput up to 3Mbps • In all existing phones, tablets, laptops	• Bluetooth® Dual Mode = Bluetooth® Classic + Bluetooth® Low Energy	• Custom profile, allows any application • Lower power consumption than Bluetooth® (down to 1/10th), multiyear on Coin Cell Battery • Throughput up to 2Mbps

Courtesy of Texas Instruments

BLE specification and features

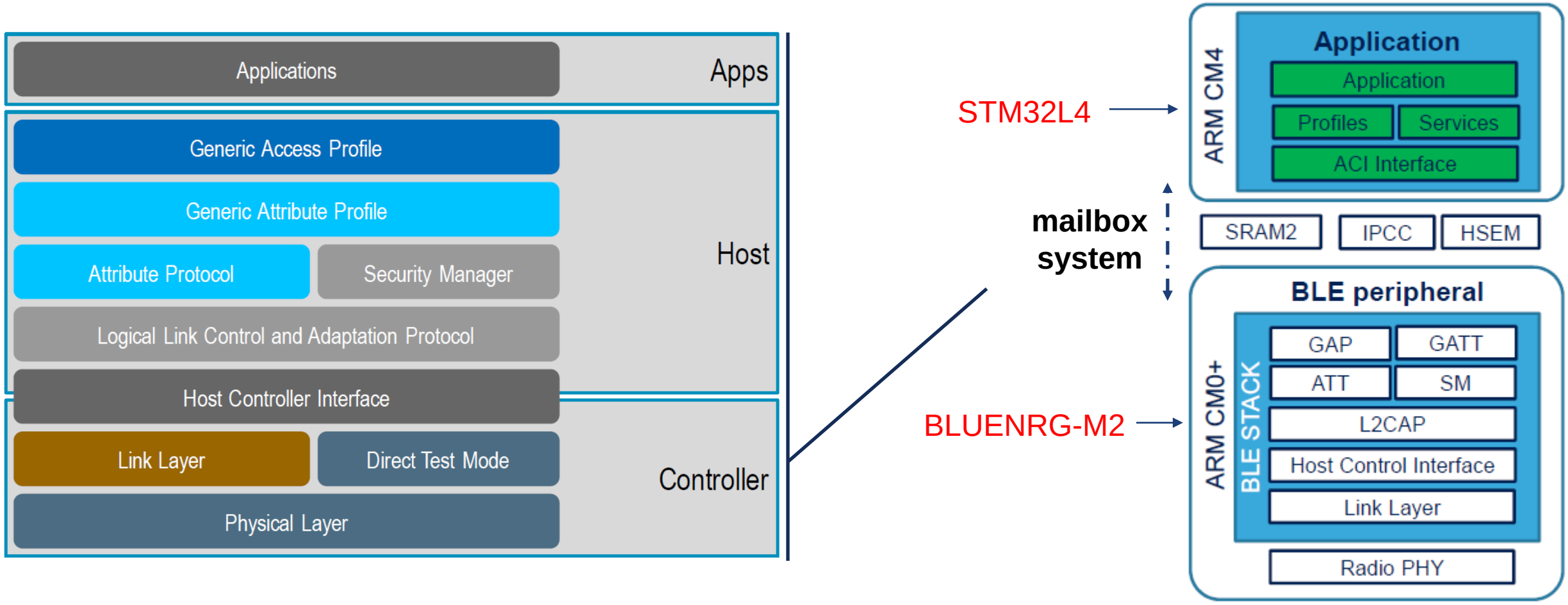
Spec	Bluetooth® 4.0	Bluetooth® 4.1	Bluetooth® 4.2	Bluetooth® 5
Speed (max)	1 Mbps	1 Mbps	1 Mbps	2 Mbps
Packet Capacity	27 bytes	27 bytes	251 bytes (2.5x faster)	251 bytes (4x faster @ 2 Mbps)
Multi-role	-	✓	✓	✓
LE Secure Connections	-	-	✓	✓
LE Privacy 1.2	-	-	✓	✓
Coded PHY Enable Longer Range (125kbps/500kbps)	-	-	-	✓
Advertisement Extension	-	-	-	✓

Courtesy of Texas Instruments

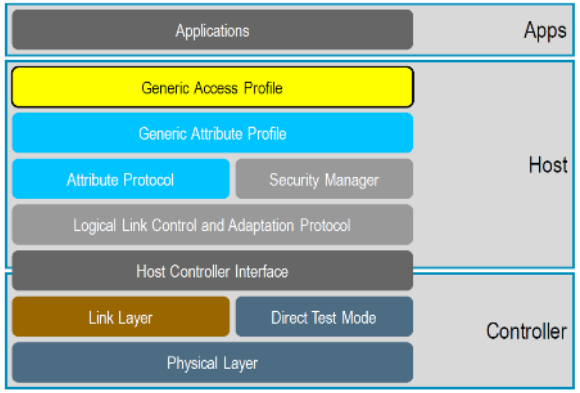
BLE foundamentals

figures from

Stack Architecture



GAP Characteristics



GAP is the lowest layer of the Bluetooth stack that an application interfaces with. It includes parameters that govern advertising and connection among other things.

Generic Access Profile (GAP):

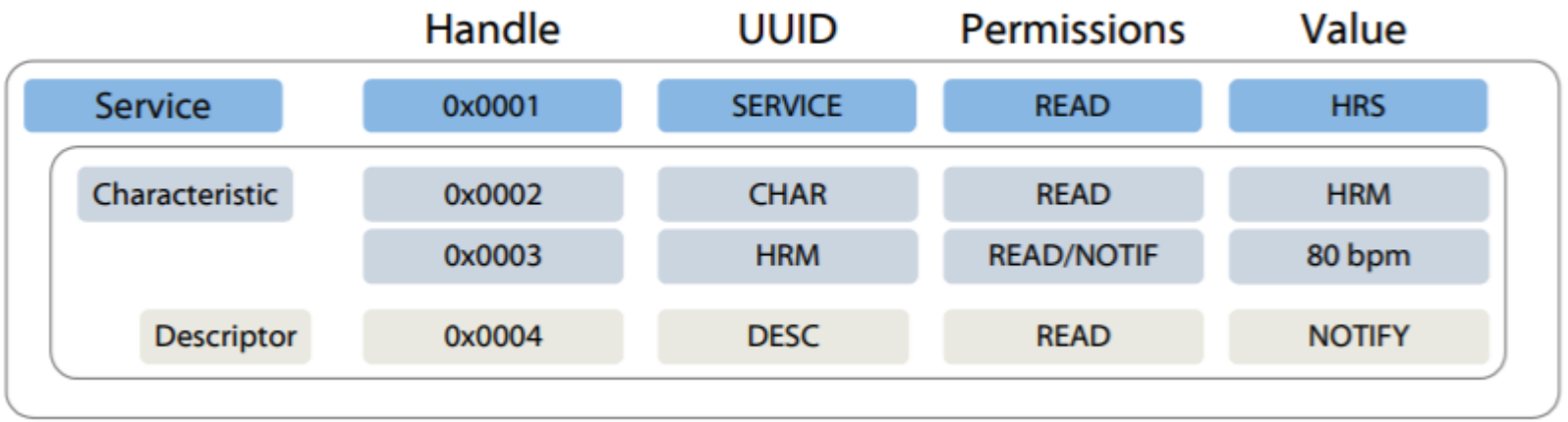
Profile Roles:

- Broadcaster, Observer, **Pheripheral, Central**
- Discoverable, Connectable, Bonding
- Privacy: Private Addresses
- 5 Link layer states: Standby, Advertising, Scanning, Initiating, One device may have one or multiple roles, working in one or multiple states at the same time.

Characteristic	Description
«Device Name»	the local name of the device
«Appearance»	enumeration of what the device "looks like"
«Peripheral Privacy Flag»	does this peripheral support privacy – is it enabled
«Reconnection Address»	address to use when reconnecting to a private device
«Peripheral Preferred Connection Parameters»	what this peripheral would prefer the central connect with

GATT – Generic Attribute Profile

Roles In addition to the roles in GAP - GATT Server and a GATT Client. The device holding data is the GATT Server, while the device accessing it is the GATT Client.
A GATT Server organizes data in what is called an attribute table and it is the attributes that contain the actual data.



GAP structure

Attribute

GATT Client

Attribute protocol (ATT)

GATT Server (Glucose profile)

Operations:
Read
Write
Indicate
Notify

Primary service Declaration
UUID: 0x1800

Device Information service
UUID: 0x180A

Glucose service
UUID: 0x1808

Glucose Measurement Characteristic
UUID: 0x2A18

Glucose Meas. Context Characteristic
UUID: 0x2A34

Glucose Feature Characteristic
UUID: 0x2A51

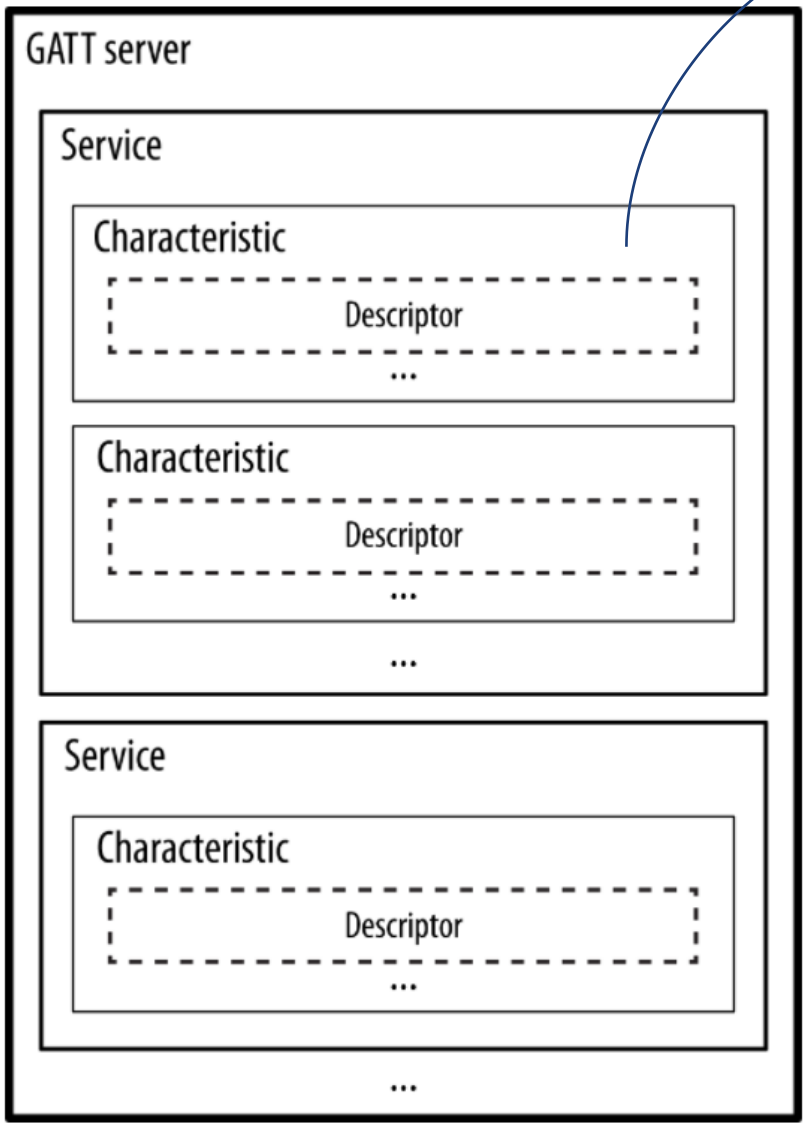
Flags (8-bit)
Sequence # (16-bit)
Carbohydrate ID (8-bit)
Carbohydrate (float)
Meal (8-bit)
Tester (nibble)
.
.
.

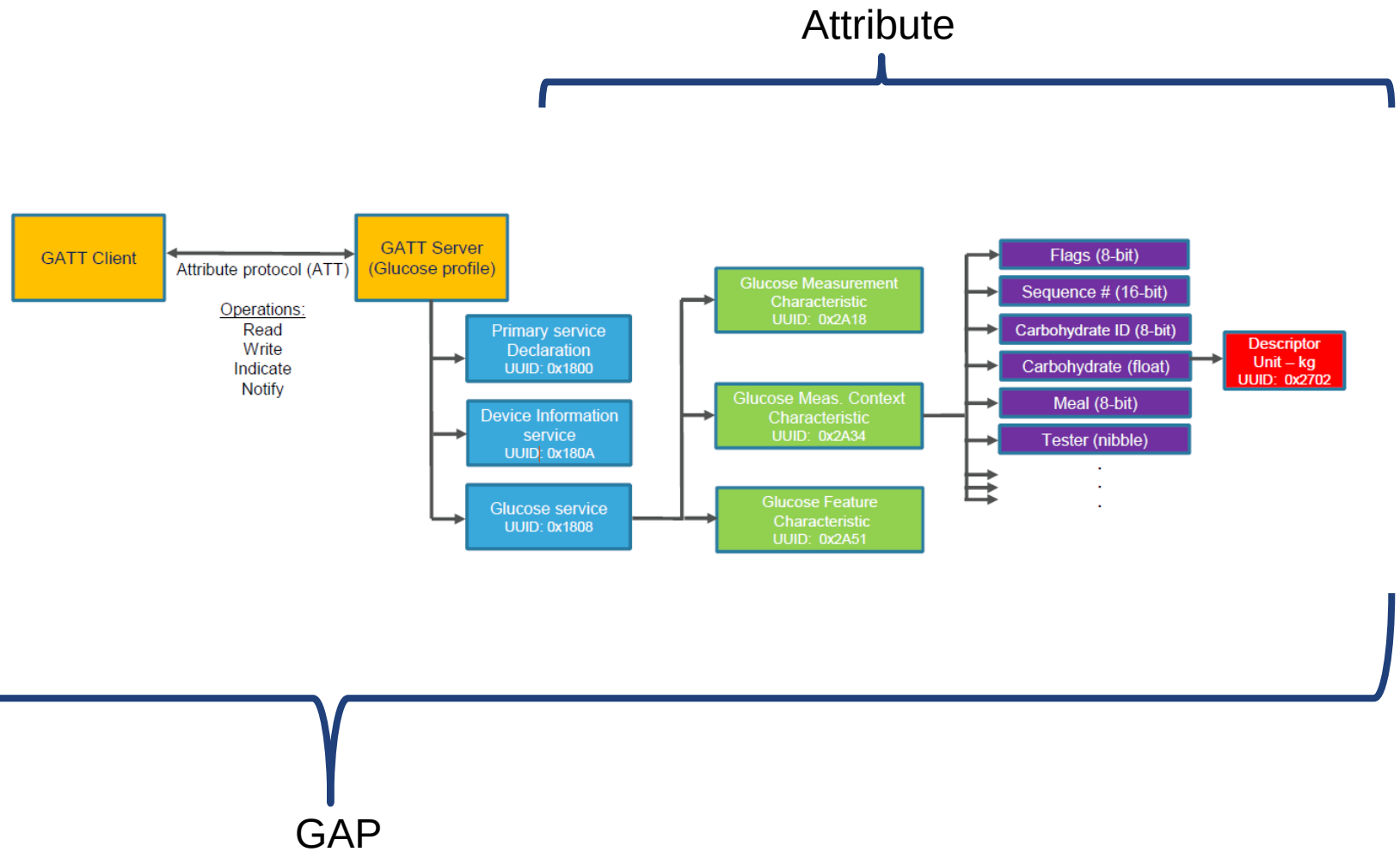
Descriptor Unit – kg
UUID: 0x2702

GAP

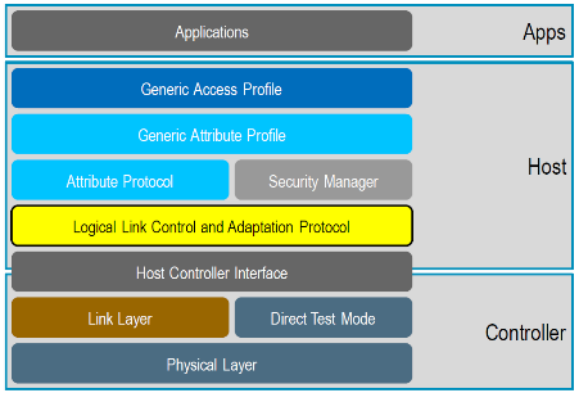
GAP structure

Handle	Type	Permissions	Value	Value length
0xNNNN	$UUID_{primary\ service}$ or $UUID_{secondary\ service}$	Read Only	Service UUID	2, 4, or 16 bytes





Topology

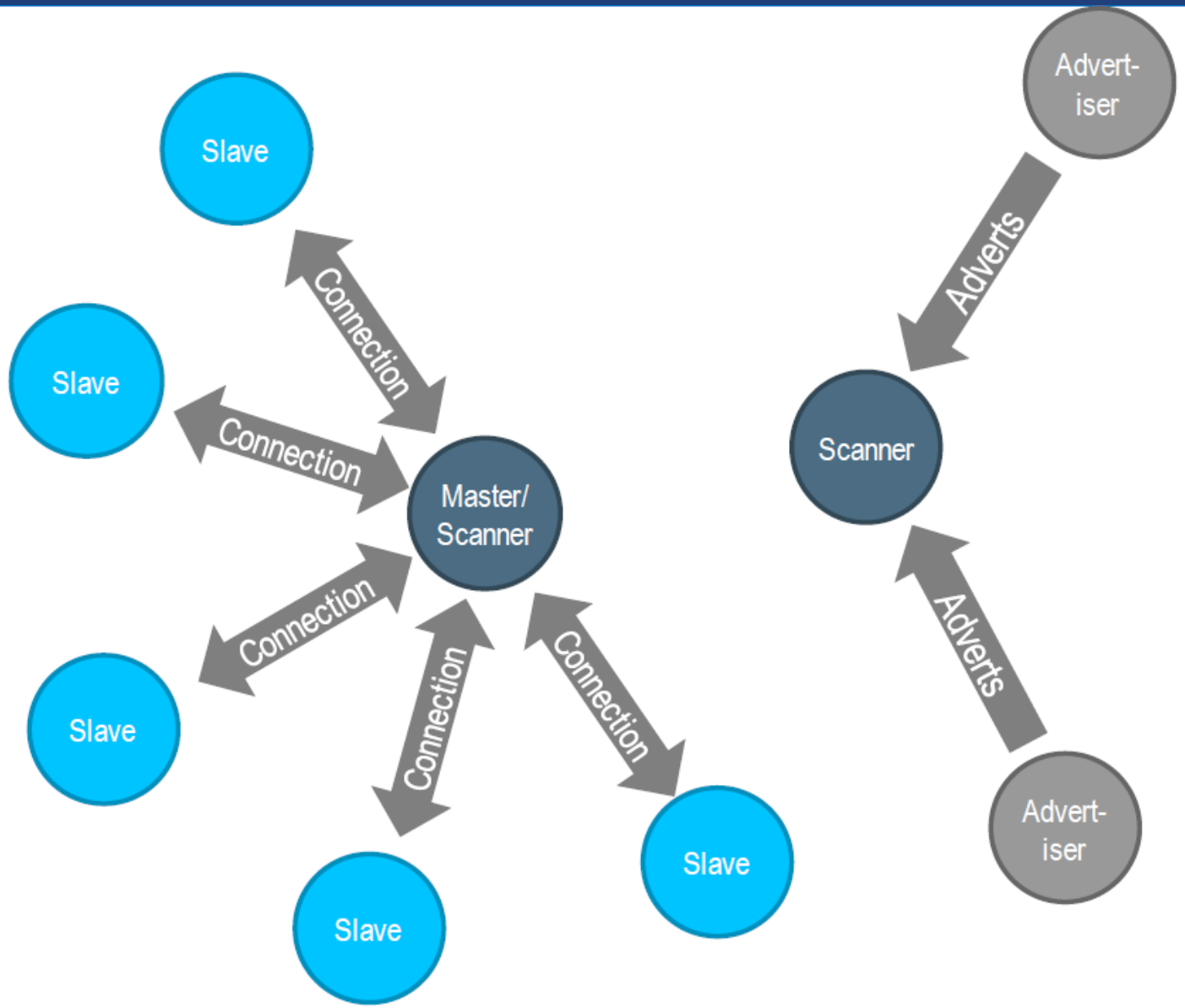


Limits:

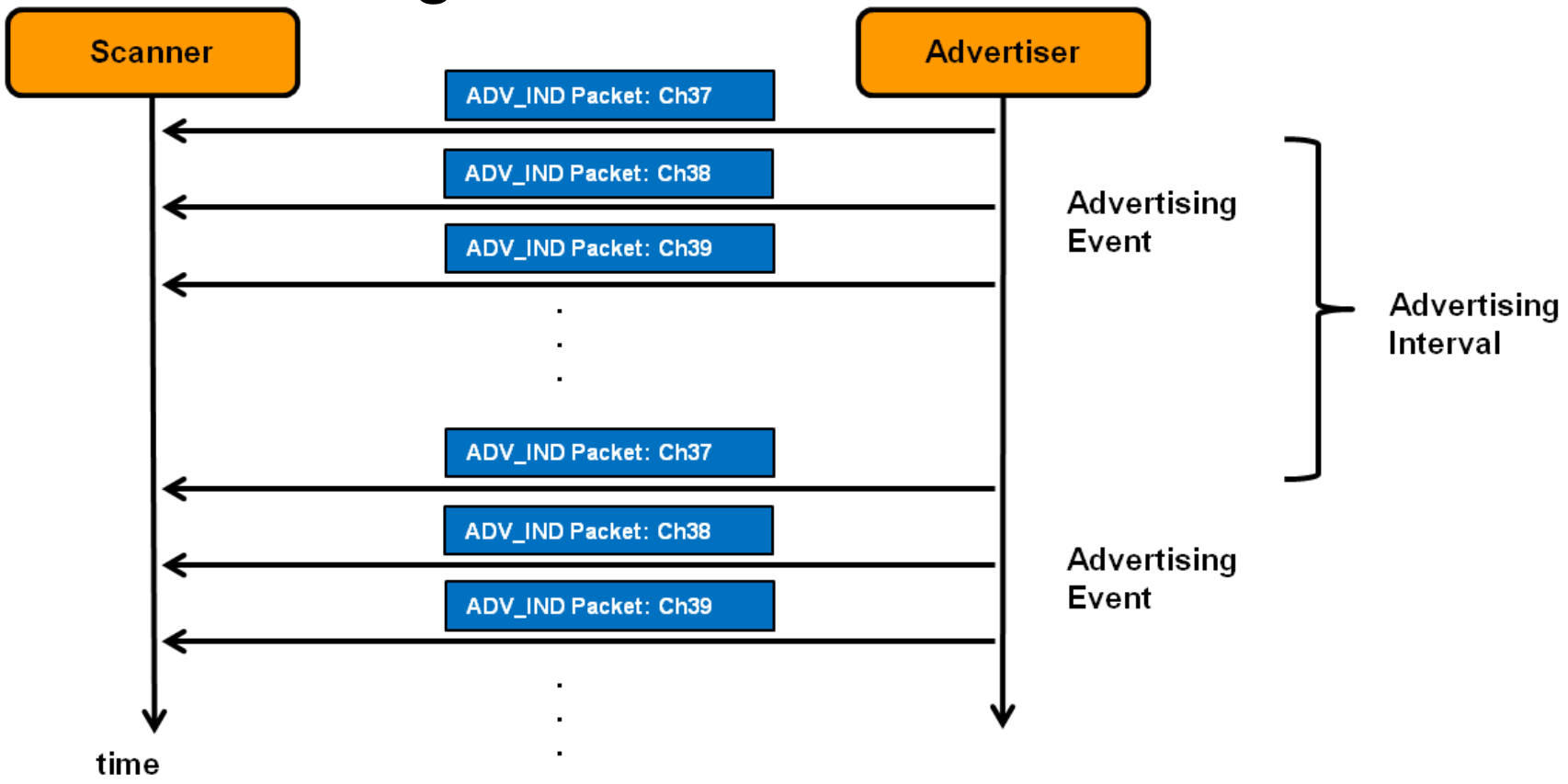
- A single master can handle ~2^31 salves
- Max. Connection Interval = 4.0 s
- ~ 800 active slaves per master

Connections:

- Used to send application data reliably and robustly
- Ultra low power connection mode
- Adaptive frequency hopping
- Connection supervision timeout



Advertising



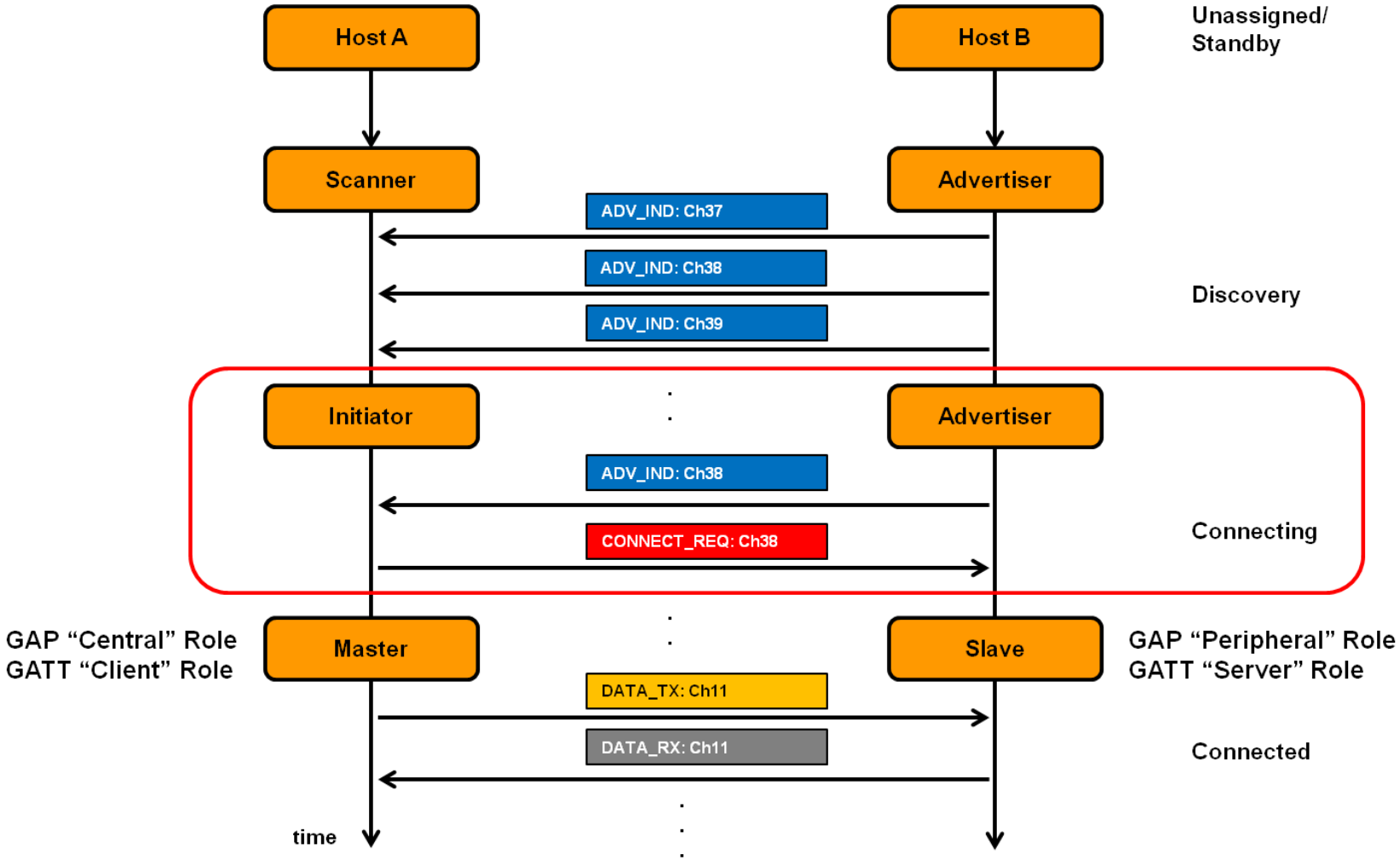
Bluetooth devices send advertising packets (PDUs) to broadcast data, and to allow other devices (scanners) to find and connect to them. The advertising data consists up to 31 bytes of user configurable data.

Interval between 20 ms to 10.24 s

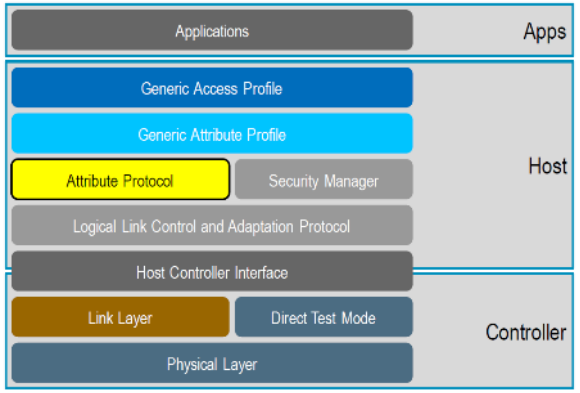
The device advertises on three channels. The advertising channels are channel 37 (2402 MHz), channel 38 (2426 MHz), and channel 39 (2480 MHz).

Connection

A connection implies a link between devices over time. BLE is a synchronous radio frequency (RF) protocol, meaning that any transmission between devices must be scheduled. A BLE connection can thus be perceived as a series of meetings where two devices transmit and receive information at the same time, on the same radio frequency. In order for this to work, the devices must agree on where (that is, on what frequency) and when next to meet.



Attribute Protocol



Attribute Protocol (ATT)

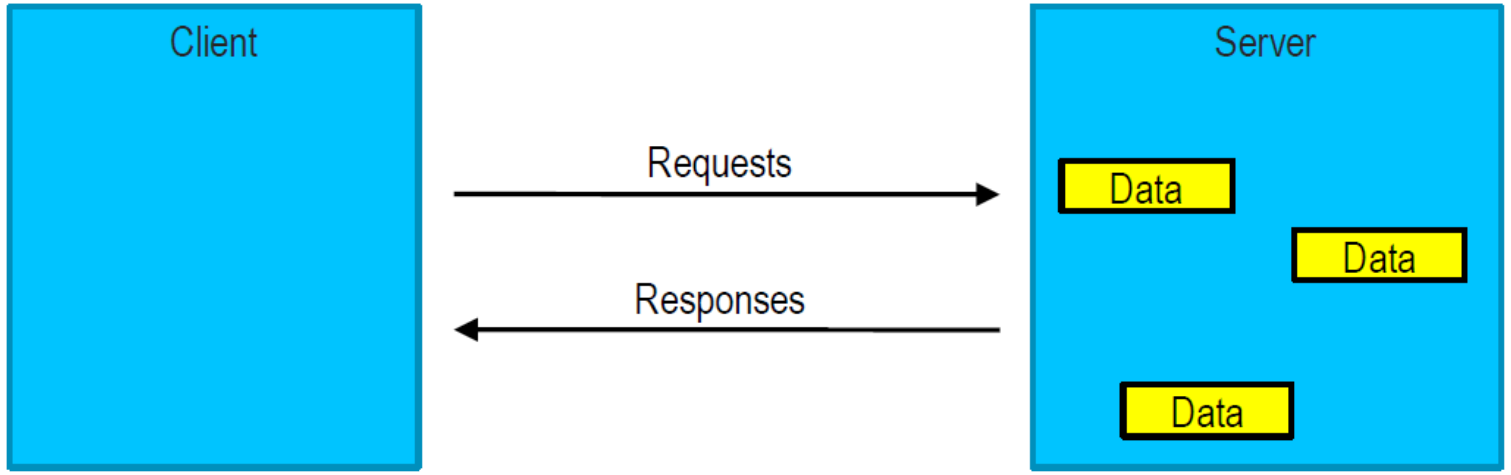
- Client Server Architecture
- Client requests data to/from servers
- Request, response, command, notification, indication, confirmation

Server have data → client wants to use this data →
Server expose Data using **Attributes**

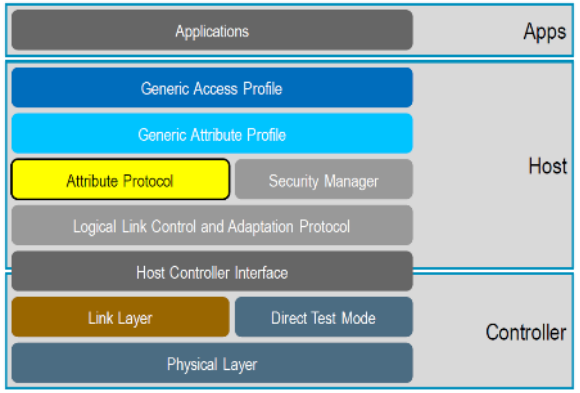
Attributes:

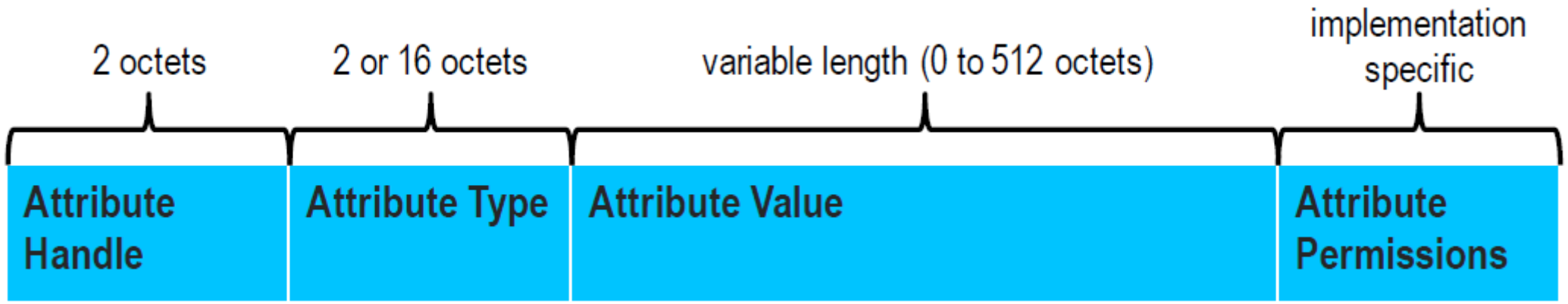
- Attributes have values (0 to 512 octets in lenght)
- Used to address an individual attribute by a client

Handle	Type	Value
0x0009	«Device Name»	0x54656d7065726174757265205365e736f72
0x0022	«Battery State»	0x04
0x0098	«Temperature»	0x0802

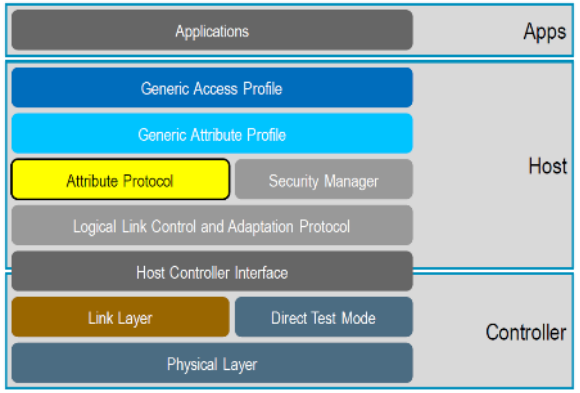


Logical Attribute Representation





Attribute Protocol



Attribute Protocol (ATT)

- Client Server Architecture
- Client requests data to/from servers
- Request, response, command, notification, indication, confirmation

Attributes:

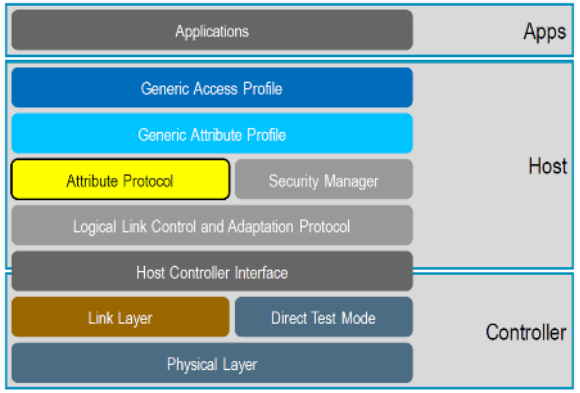
- Type: UUID → determines what the value means
- Types are defined by “Characteristic Specification” or Generic Access Profile or Generic Attribute Profile

Handle	Type	Value
0x0009	«Device Name»	“Temperature Sensor”
0x0022	«Battery State»	0x04
0x0098	«Temperature»	0x0802

- 0x54656d70657261747572652053656e736f72 = “Temperature Sensor”
- 0x04 = Discharging
- 0x0802 = 2050 * 0.01 ºC = 20.5 ºC

Handle	Type	Value
0x0009	«Device Name»	0x54656d70657261747572652053656e736f72
0x0022	«Battery State»	0x04
0x0098	«Temperature»	0x0802

Attribute Type: UUID



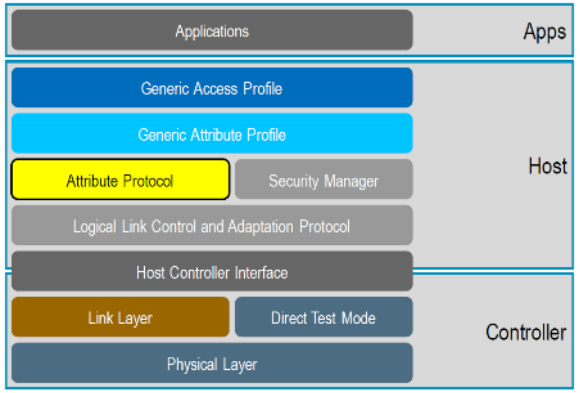
Type is a «UUID»

UUIDs are 128 bits in length

Bluetooth defines a Bluetooth Base UUID allowing a 16 bit «UUID» to be defined

Example:
00000000-0000-1000-8000-00805F9B34FB

Attribute Type: UUID



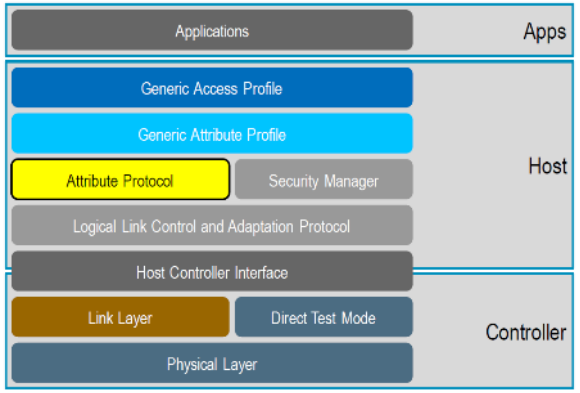
Type is a «UUID»

UUIDs are 128 bits in length

Bluetooth defines a Bluetooth Base UUID allowing a 16 bit «UUID» to be defined

Example:
0000xxxx-0000-1000-8000-00805F9B34FB
0000**1234**-0000-1000-8000-00805F9B34FB
= 16 bit UUID **0x1234**

Attribute Permissions



Attributes values may be:

- readable / not readable
- writeable / not writeable
- readable & writeable / not readable & not writeable

Attribute values may require:

- authentication to read / write
- authorization to read / write
- encryption / pairing with sufficient strength to read / write

If request to read an attribute value that cannot be read

Error Response «Read Not Permitted»

If request to write an attribute value that requires authentication

Error Response «Insufficient Authentication»

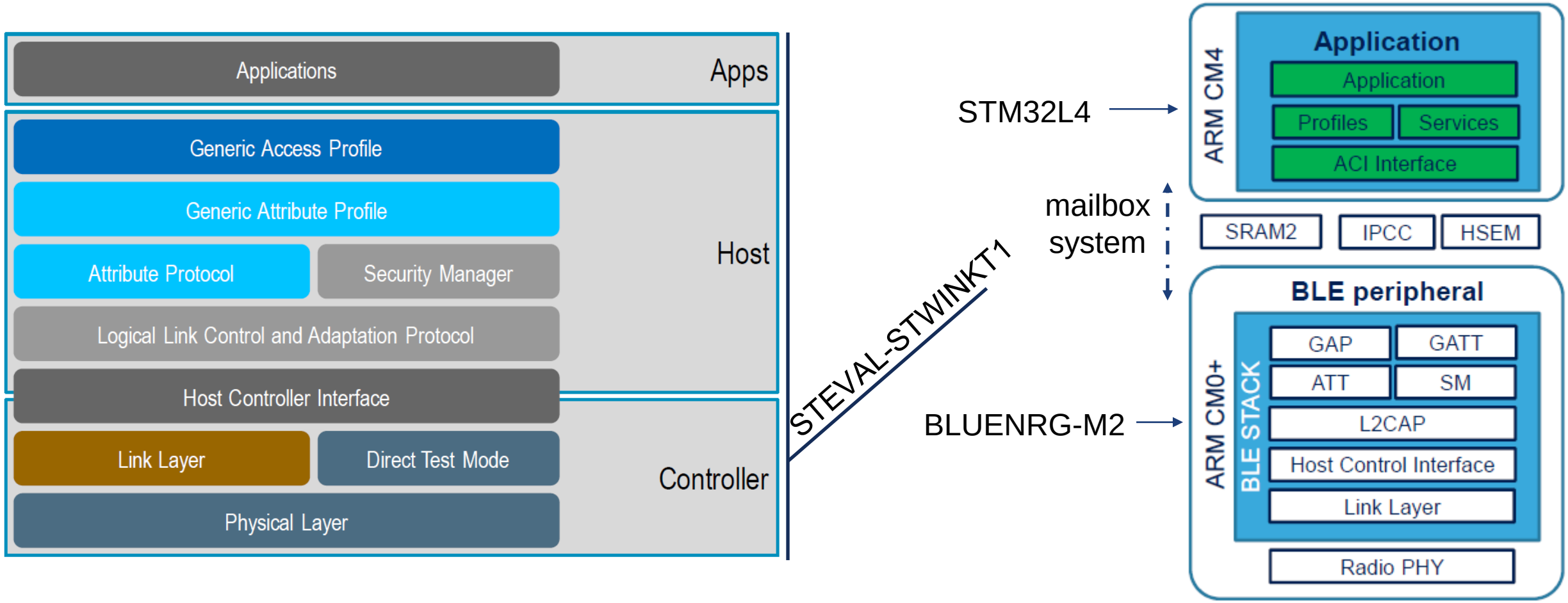
Client must create authenticated connection and then retry

There is no "pending" state

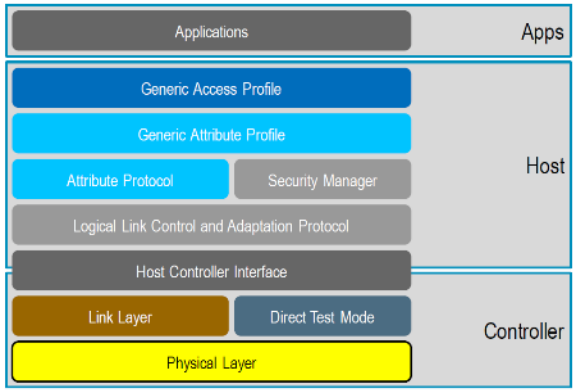
BLE Appendix

figures from

Stack Architecture

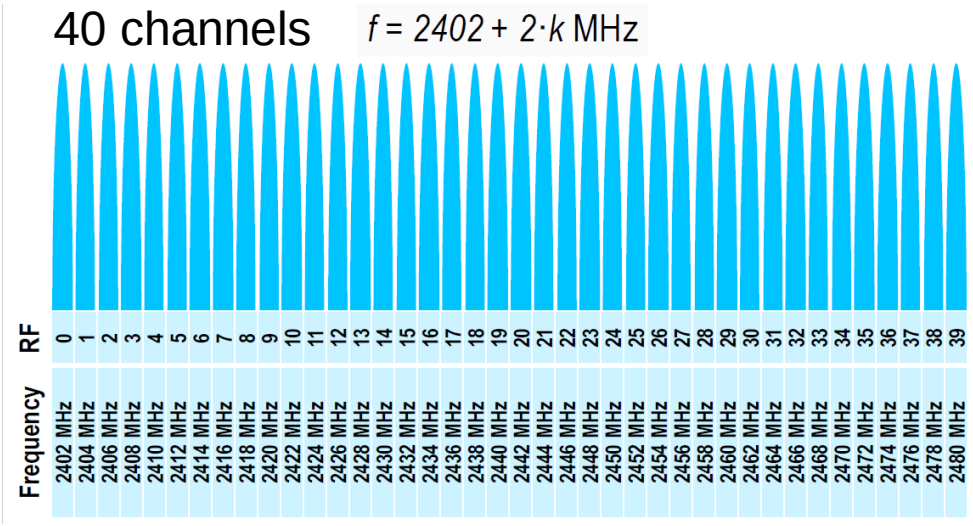


Physical Layer



Uses 2.4 GHz ISM Band

- Industrial Scientific Medical band
- License Free between 2400 MHz and 2483.5 MHz
- Used by IEEE 802.11, IEEE 802.15.4 and many other proprietary radios



2 MHz channel spacing

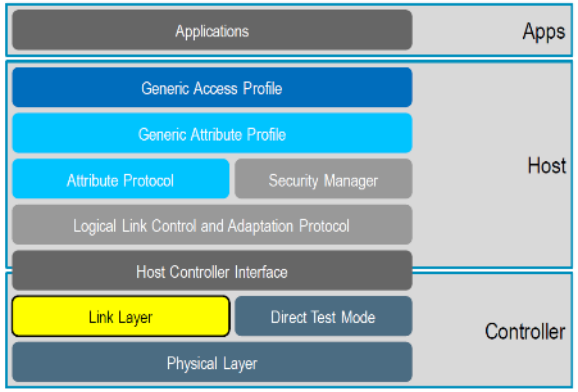
GFSK Modulation

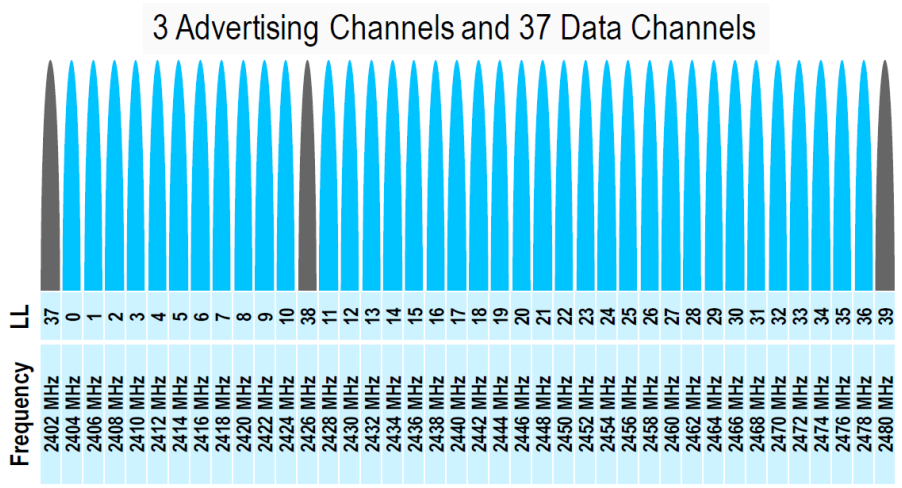
- Bit period product = 0.5
- Modulation index = 0.5
- Bandwidth = 1 Mbps

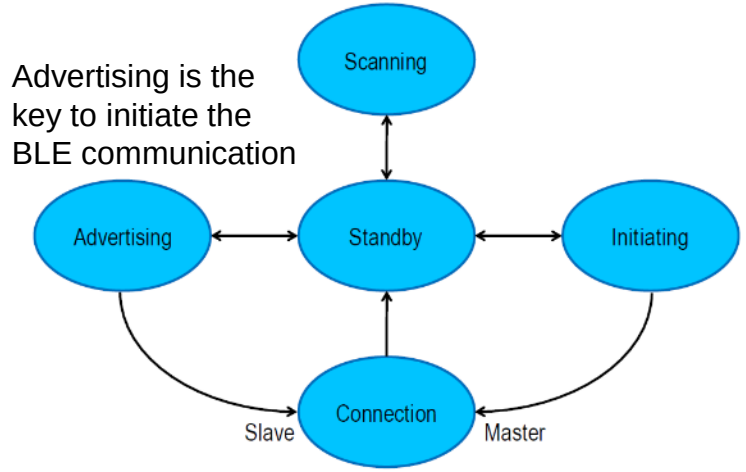
Why GFSK ?

- “pulse shaping”
- Reduced spectral width

Link Layer

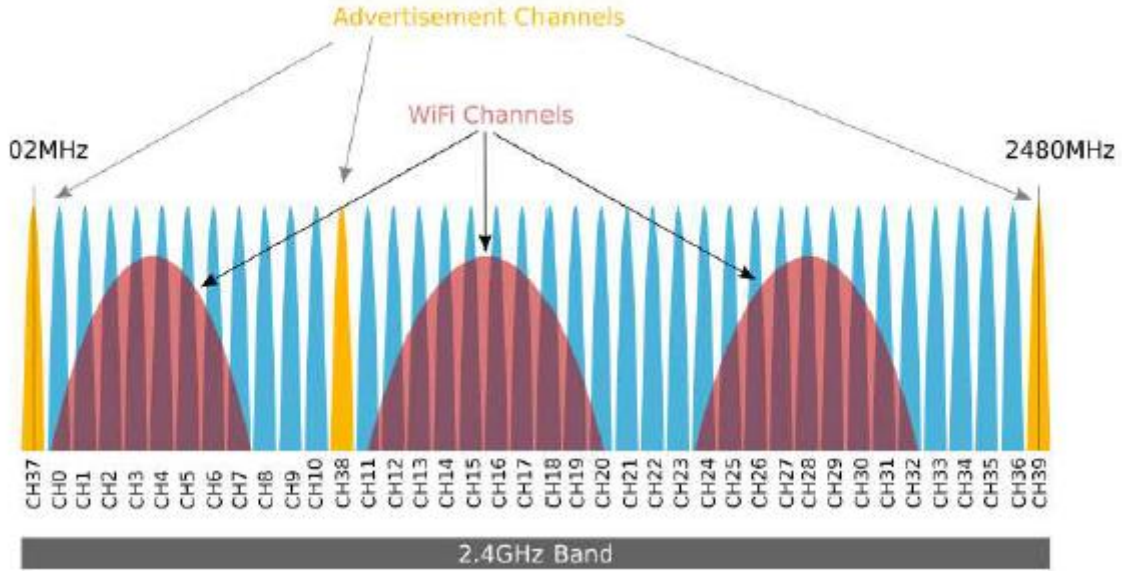




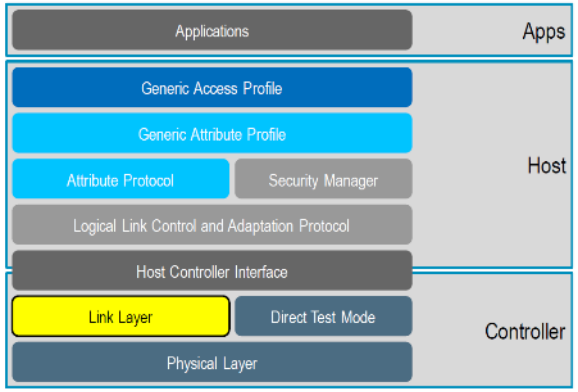


Link Layer (LL)

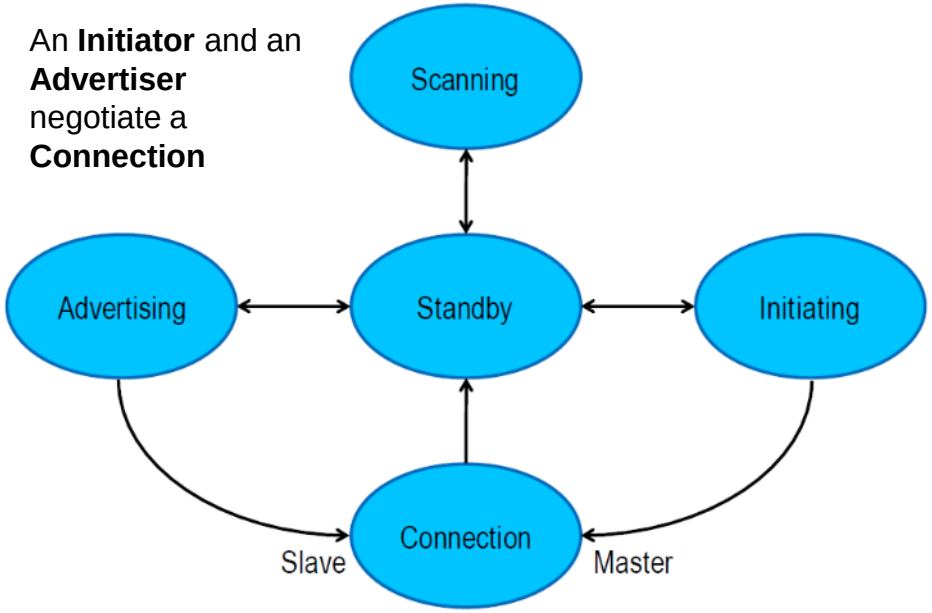
- Advertising Channels
- Data Channels
- Control Procedures
- Defines packet structure
- One state machine per conn. device
- Link-layer-level encryption



Link Layer



An **Initiator** and an **Advertiser** negotiate a **Connection**



In a connection:

- The link-layer **Master** is also the GAP **Central**
- The link-layer **Slave** is also the GAP **Peripheral**

Host Control Interface (HCI)

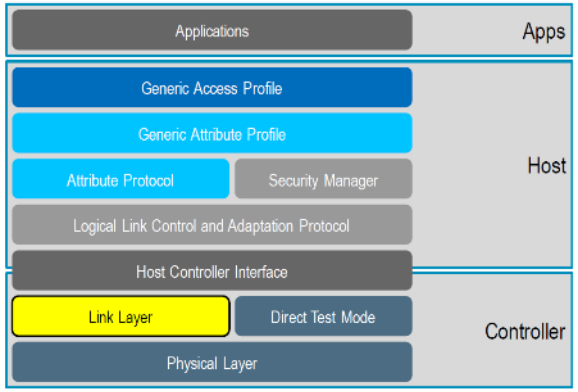
- Bridge between radio and MCU domain

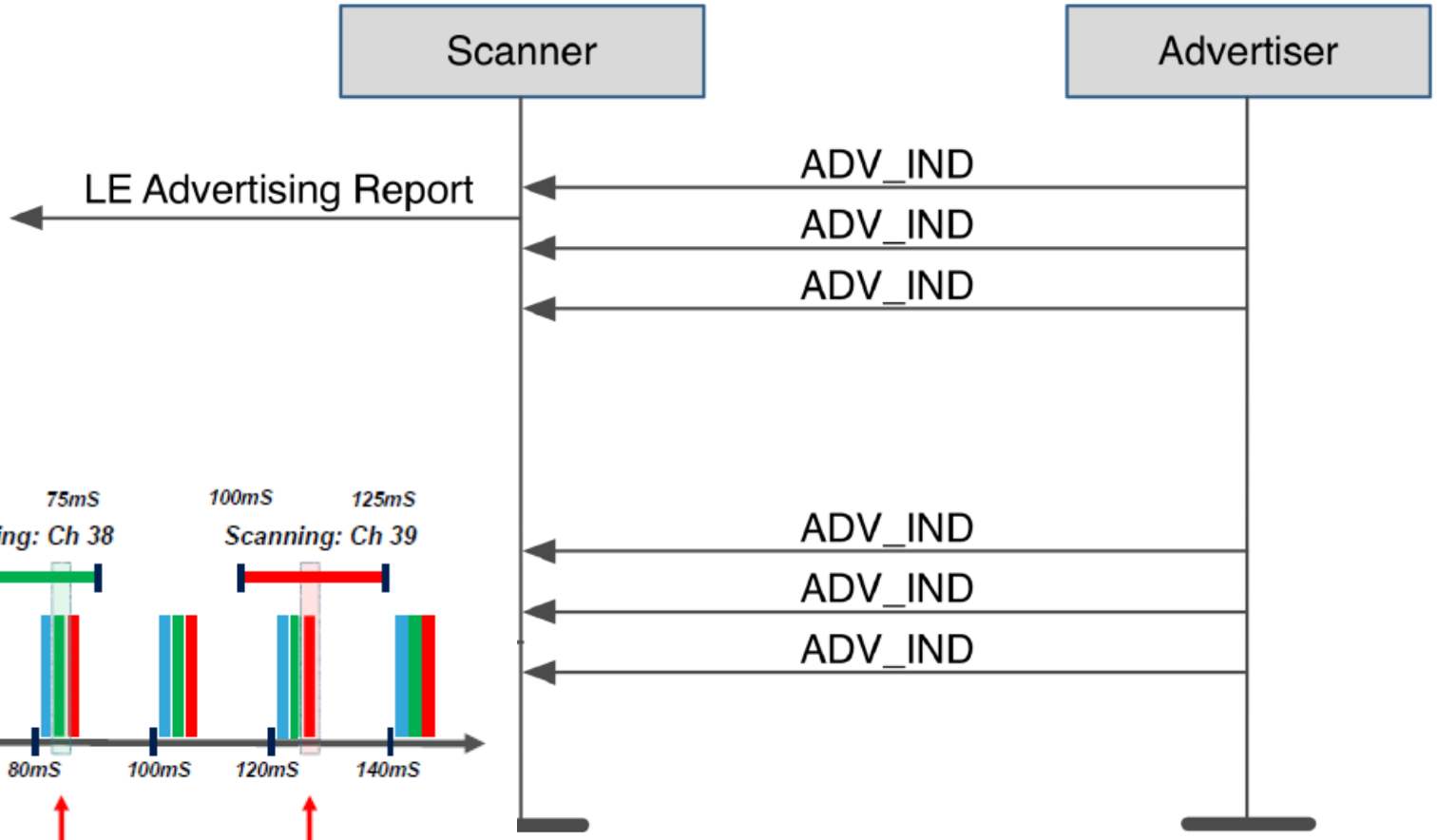
L2CAP

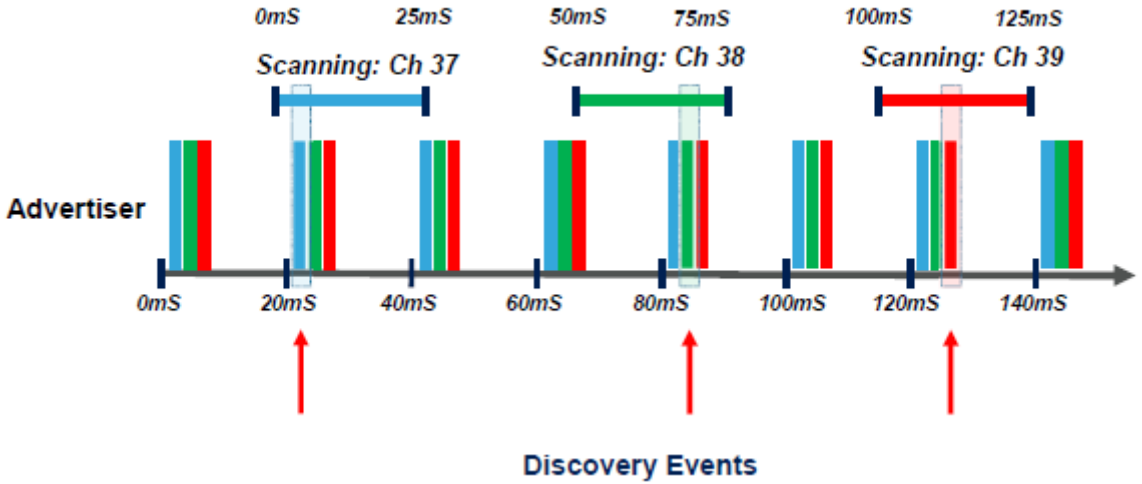
(Logical Link Control and Adaption Control Control Procedures)

- Multiplex packets from higher-level protocols
- Handles segmentations and reassembly of packets
- Quality of Services

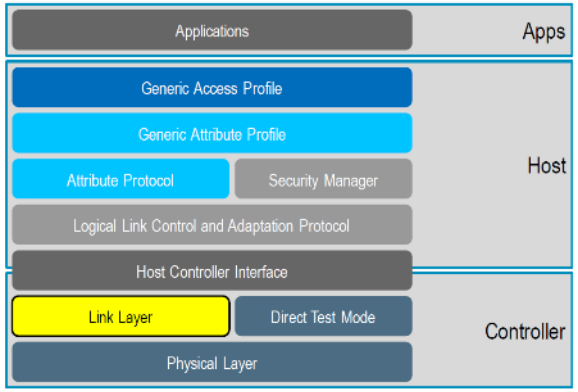
Passive Scanning







Passive Scanning

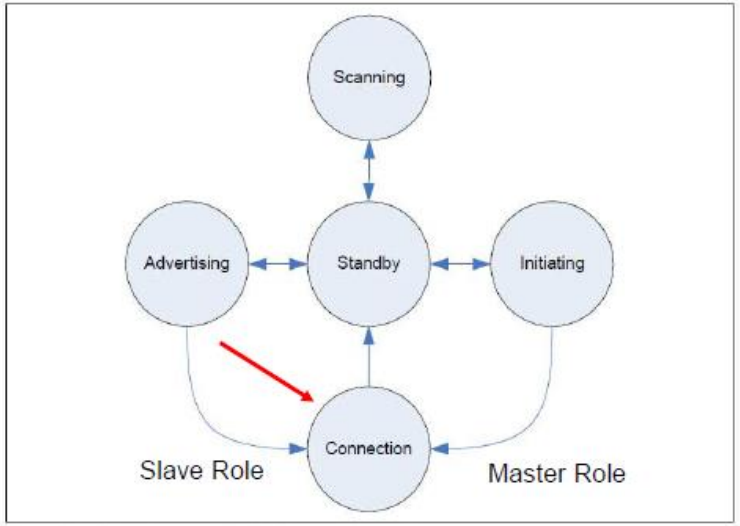


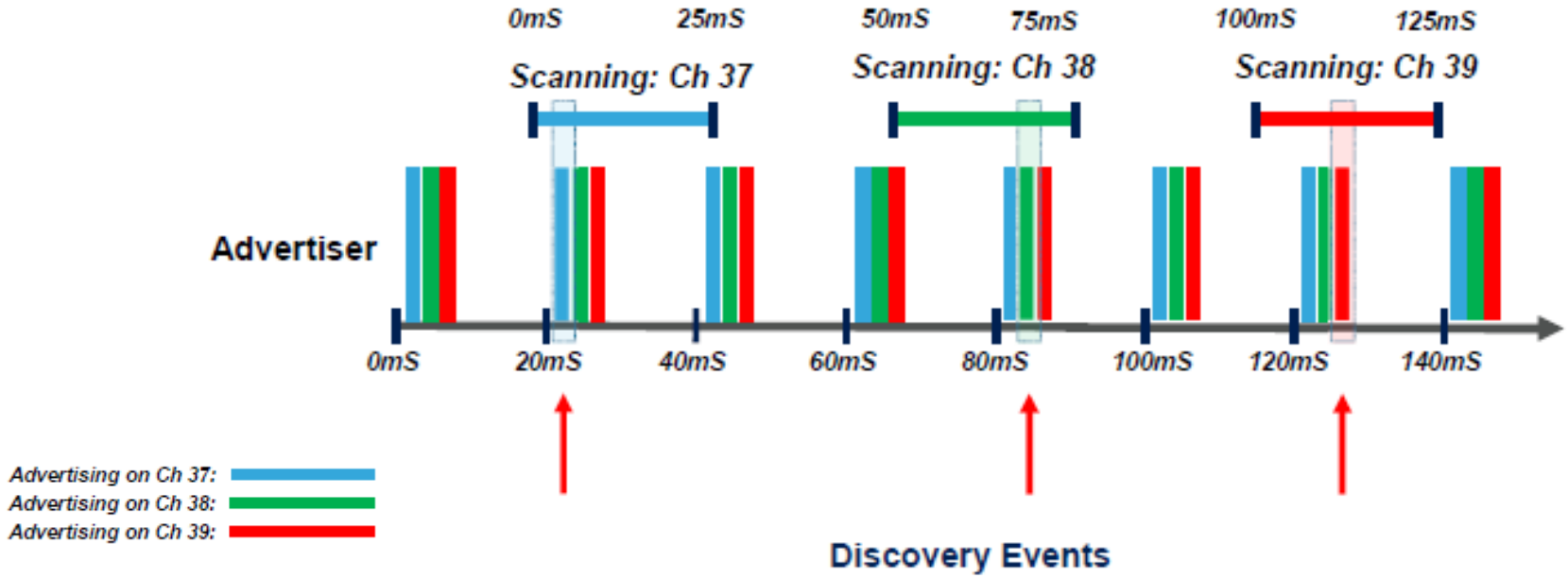
Advertiser Settings:

- Advertising Interval: 20 ms

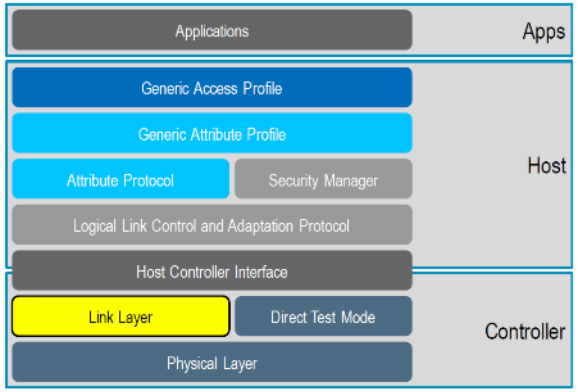
Scanner Settings:

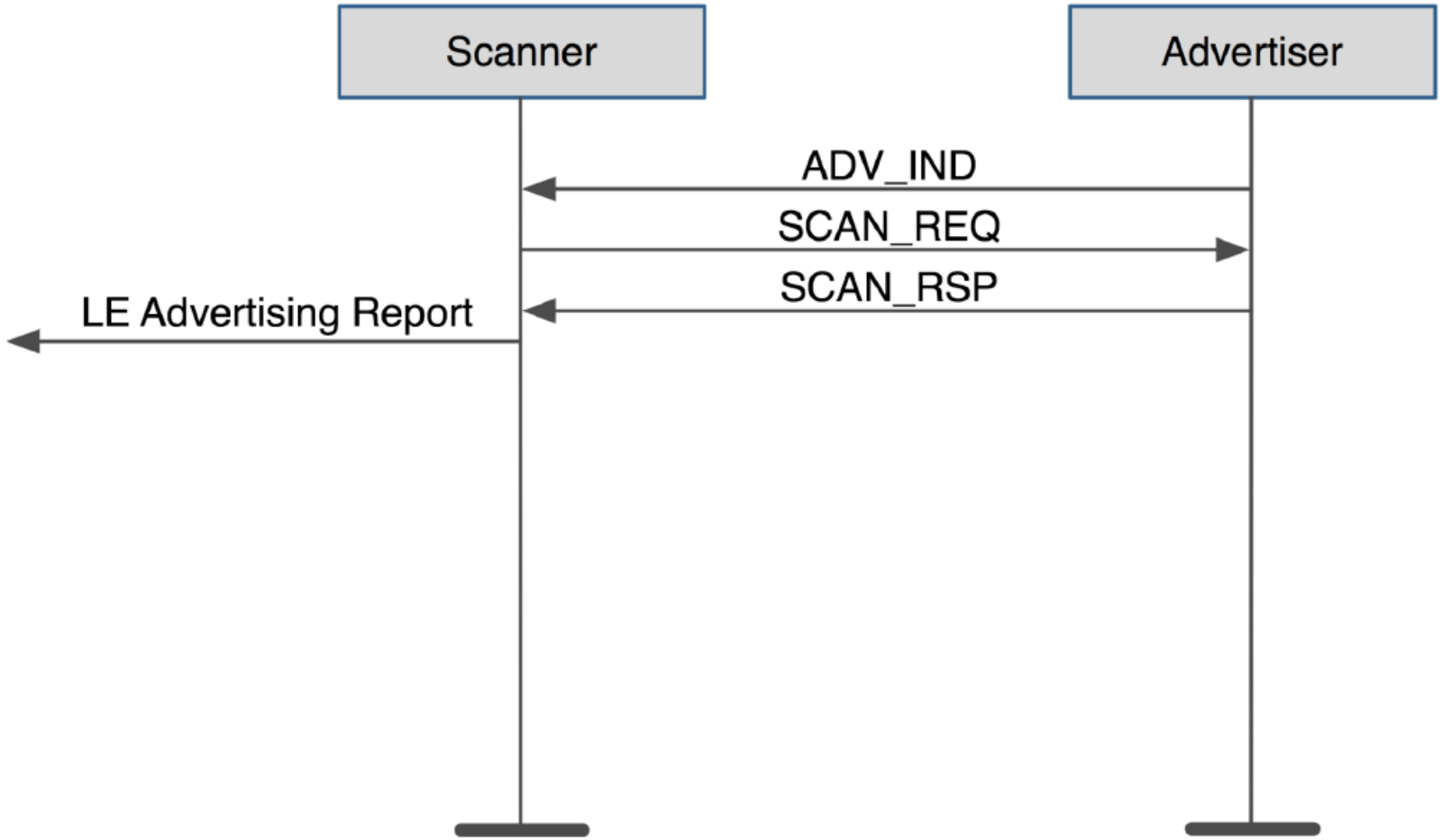
- Scan Interval: 50 ms
- Scan Windows: 25 ms



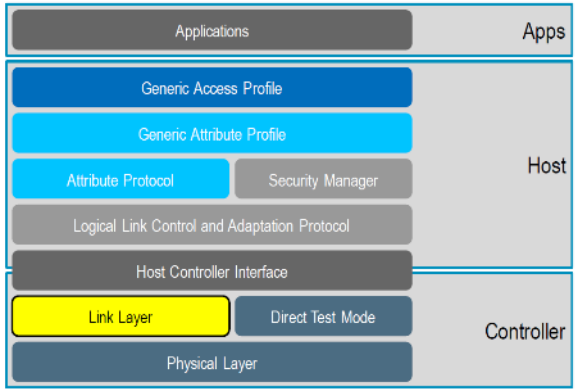


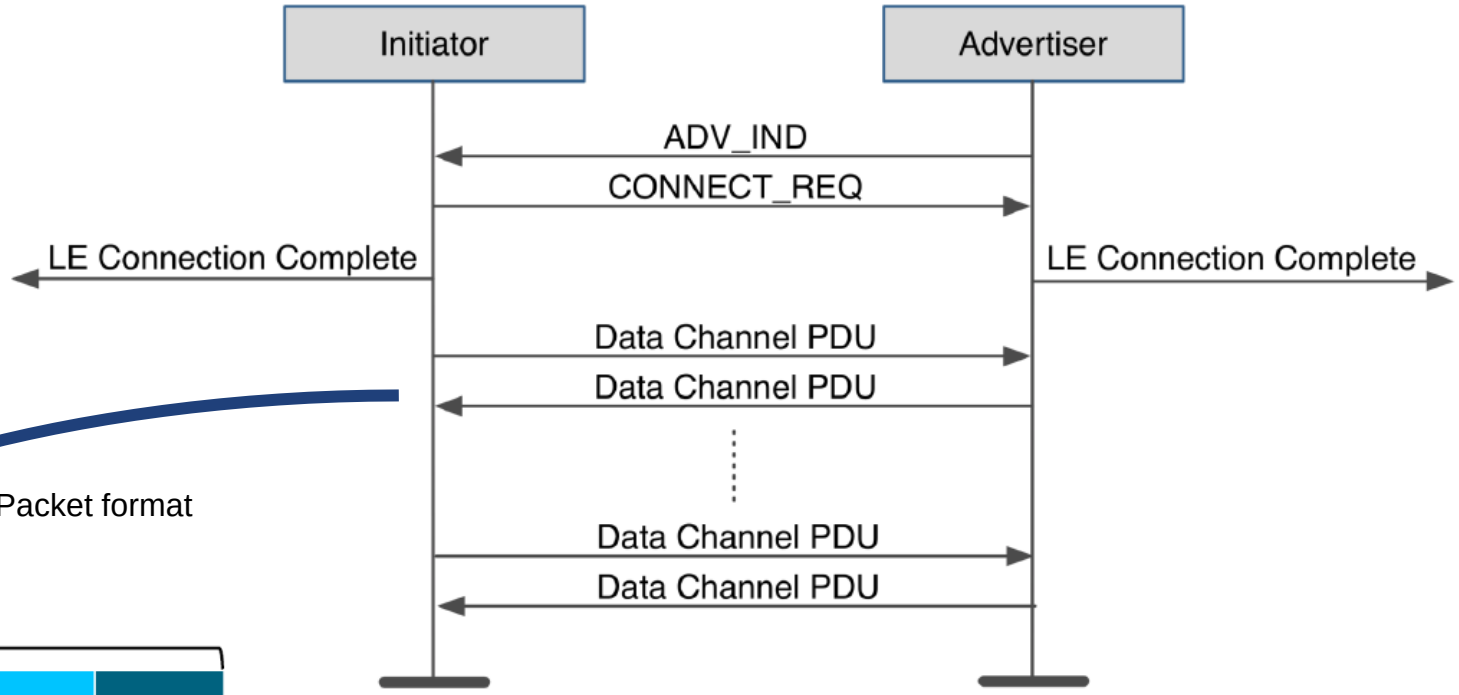
Active Scanning



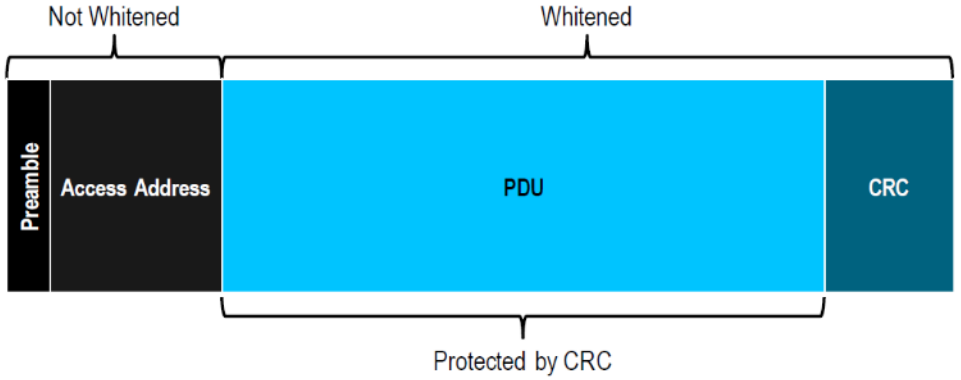


Initiating Connections

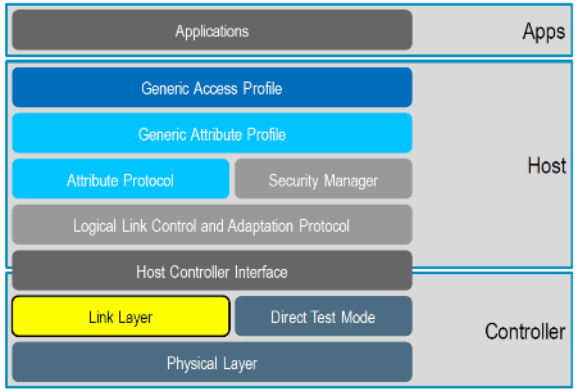


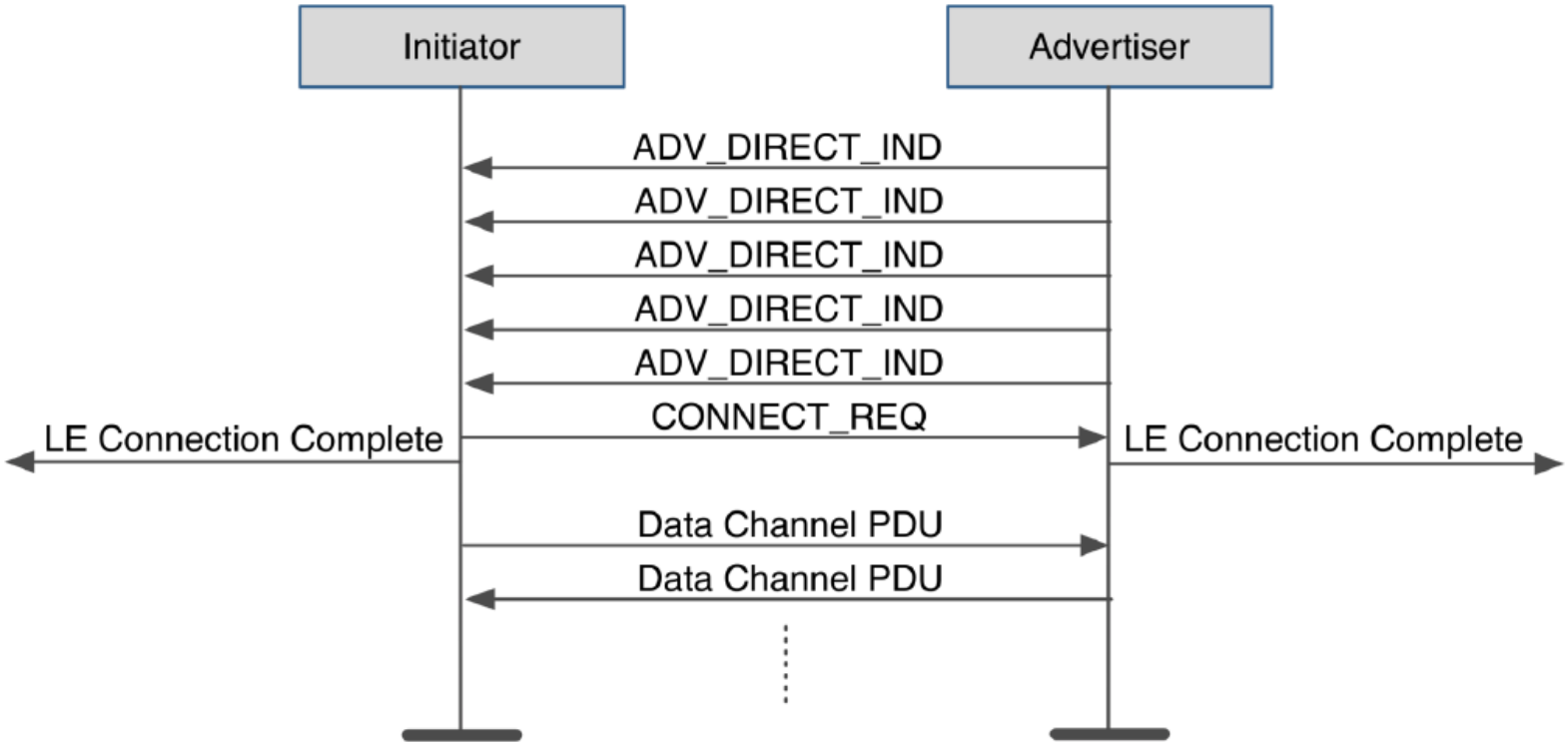


PDU Packet format

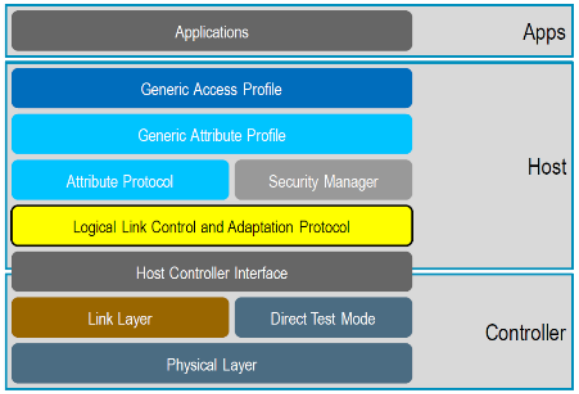


Directed Connection





Topology

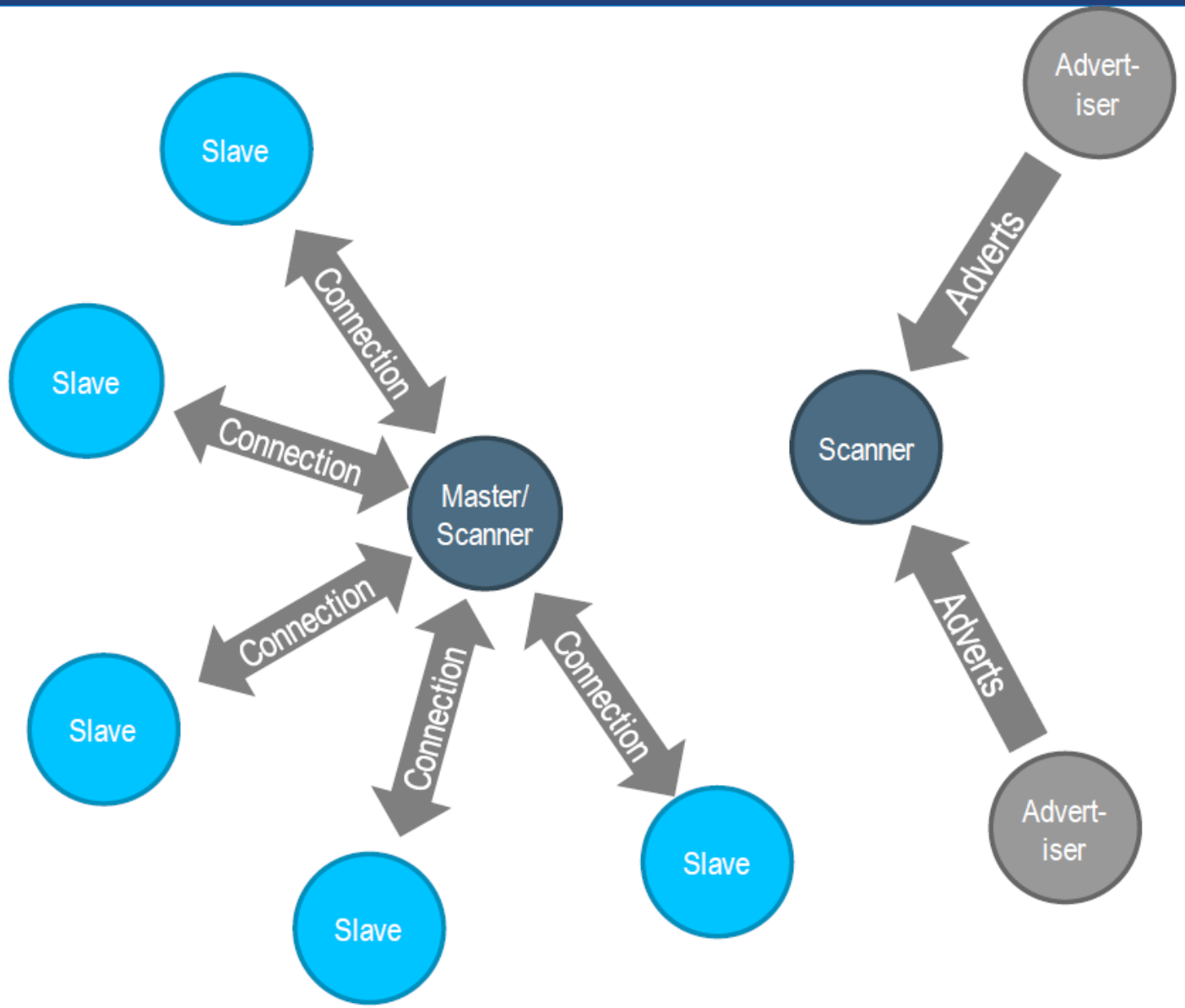


Limits:

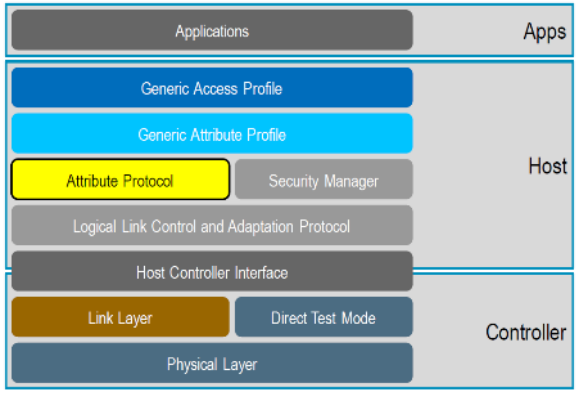
- A single master can handle ~2^31 salves
- Max. Connection Interval = 4.0 s
- ~ 800 active slaves per master

Connections:

- Used to send application data reliably and robustly
- Ultra low power connection mode
- Adaptive frequency hopping
- Connection supervision timeout



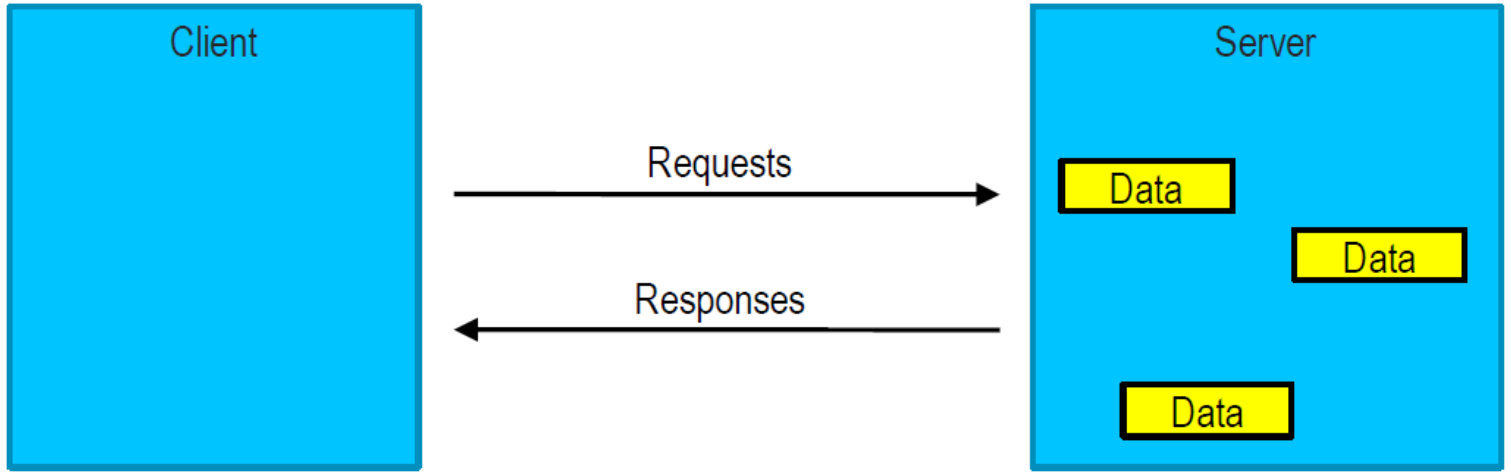
Attribute Protocol



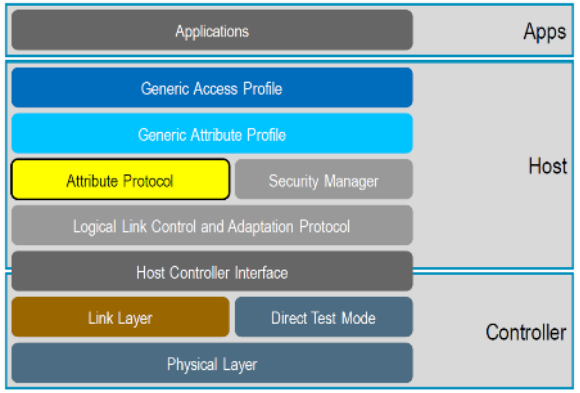
Server have data → client wants to use this data →
Server expose Data using **Attributes**

Attribute Protocol (ATT)

- Client Server Architecture
- Client requests data to/from servers
- Request, response, command, notification, indication, confirmation



Attribute Protocol



Server have data → client wants to use this data →
Server expose Data using **Attributes**

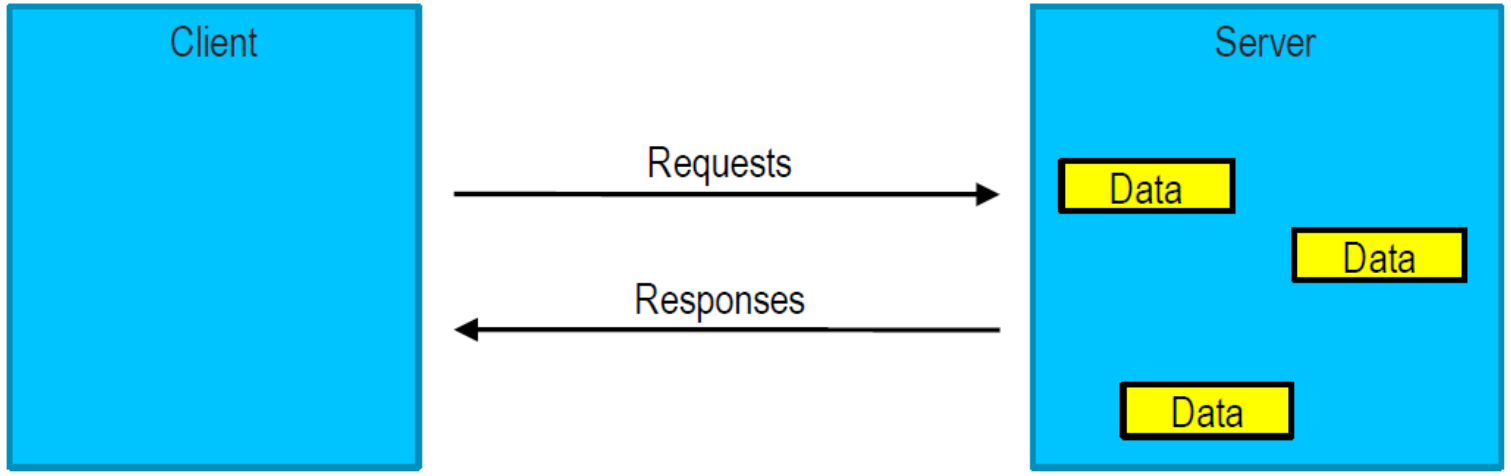
Attributes:

- Attributes have values (0 to 512 octets in lenght)
- Used to address an individual attribute by a client

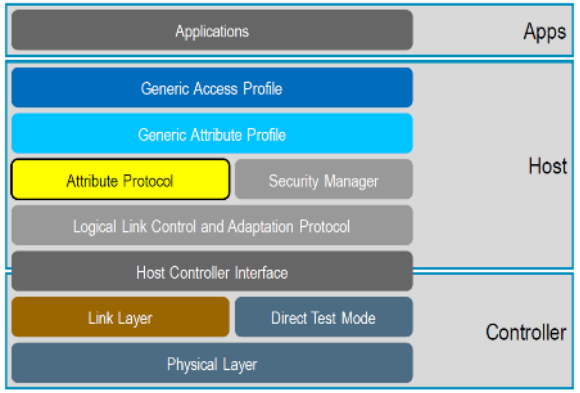
Handle	Type	Value
0x0009	«Device Name»	0x54656d7065726174757265205365e736f72
0x0022	«Battery State»	0x04
0x0098	«Temperature»	0x0802

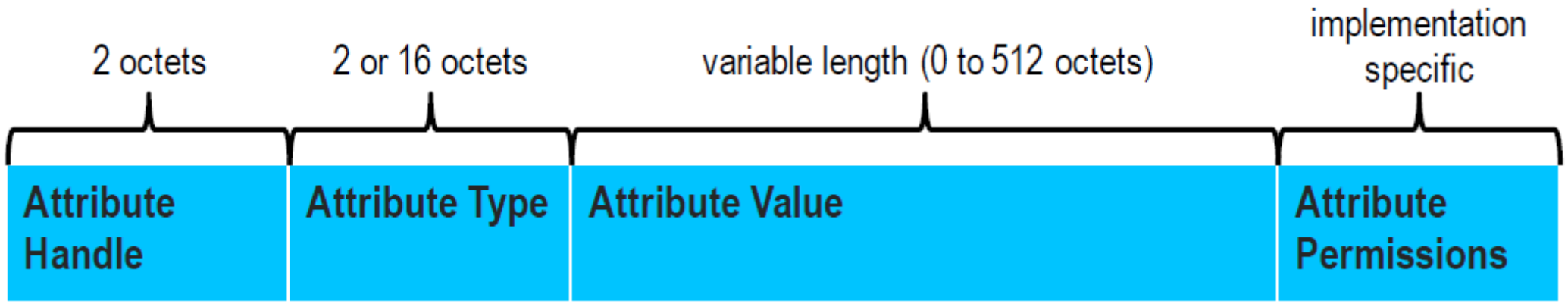
Attribute Protocol (ATT)

- Client Server Architecture
- Client requests data to/from servers
- Request, response, command, notification, indication, confirmation

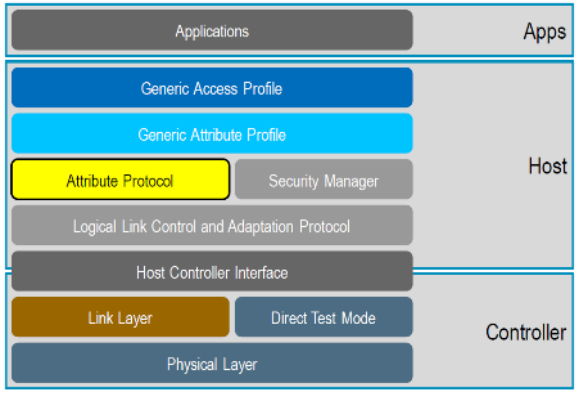


Logical Attribute Representation





Attribute Protocol



Attribute Protocol (ATT)

- Client Server Architecture
- Client requests data to/from servers
- Request, response, command, notification, indication, confirmation

Attributes:

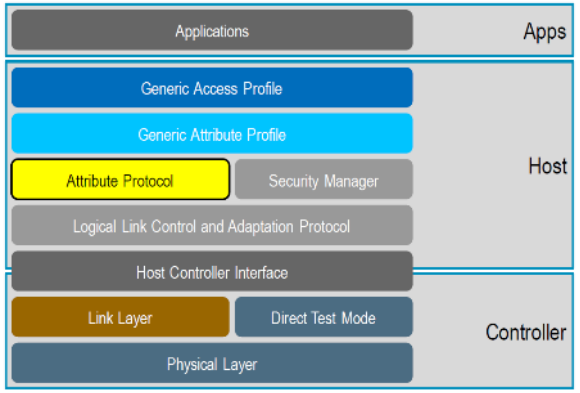
- Type: UUID → determines what the value means
- Types are defined by “Characteristic Specification” or Generic Access Profile or Generic Attribute Profile

Handle	Type	Value
0x0009	«Device Name»	“Temperature Sensor”
0x0022	«Battery State»	0x04
0x0098	«Temperature»	0x0802

- 0x54656d70657261747572652053656e736f72 = “Temperature Sensor”
- 0x04 = Discharging
- 0x0802 = 2050 * 0.01 ºC = 20.5 ºC

Handle	Type	Value
0x0009	«Device Name»	0x54656d70657261747572652053656e736f72
0x0022	«Battery State»	0x04
0x0098	«Temperature»	0x0802

Attribute Type: UUID



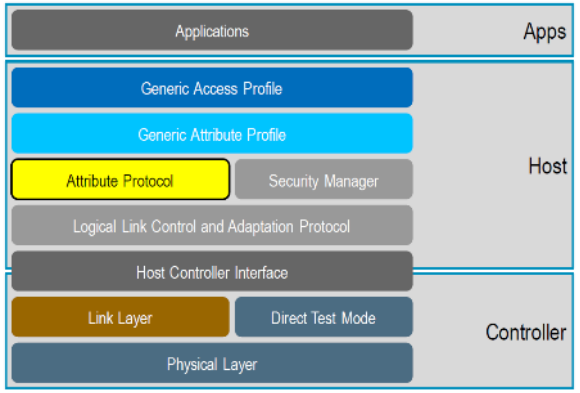
Type is a «UUID»

UUIDs are 128 bits in length

Bluetooth defines a Bluetooth Base UUID allowing a 16 bit «UUID» to be defined

Example:
00000000-0000-1000-8000-00805F9B34FB

Attribute Type: UUID



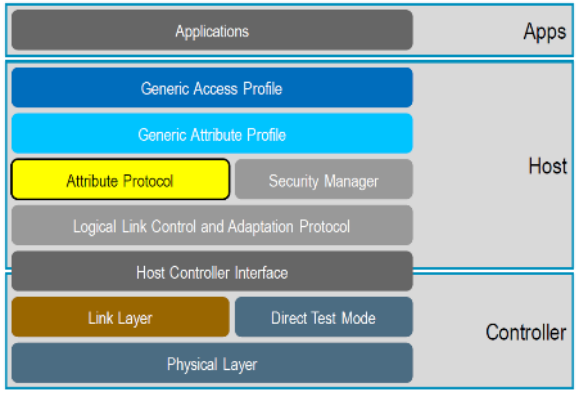
Type is a «UUID»

UUIDs are 128 bits in length

Bluetooth defines a Bluetooth Base UUID allowing a 16 bit «UUID» to be defined

Example:
0000xxxx-0000-1000-8000-00805F9B34FB
0000**1234**-0000-1000-8000-00805F9B34FB
= 16 bit UUID **0x1234**

Attribute Handle



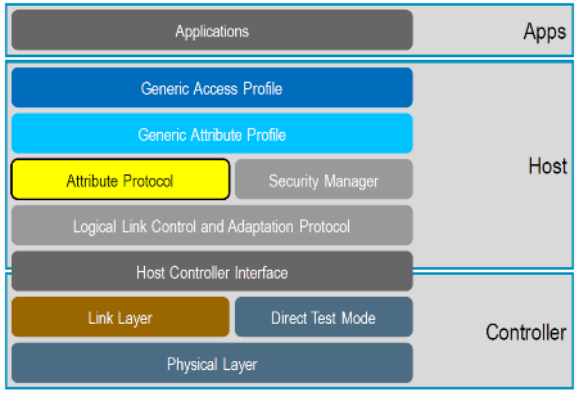
Handle is a 16 bit value

- 0x0000 is reserved – shall never be used
- 0x0001 to 0xFFFF can be assigned to any attributes

Handles are “sequential”

- 0x0005 is “before” 0x0006
- 0x0104 is “after” 0x00F8

Attribute Permissions



Attributes values may be:

- readable / not readable
- writeable / not writeable
- readable & writeable / not readable & not writeable

Attribute values may require:

- authentication to read / write
- authorization to read / write
- encryption / pairing with sufficient strength to read / write

If request to read an attribute value that cannot be read

Error Response «Read Not Permitted»

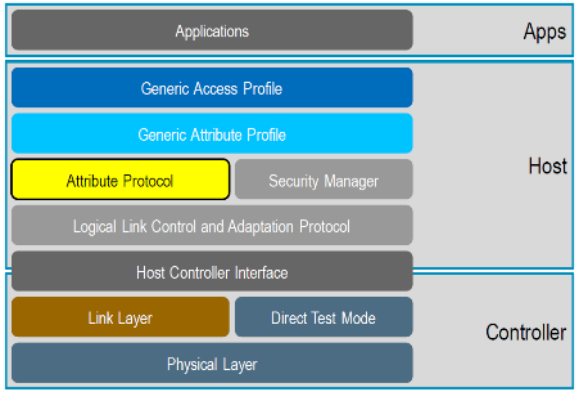
If request to write an attribute value that requires authentication

Error Response «Insufficient Authentication»

Client must create authenticated connection and then retry

There is no "pending" state

Attribute Permissions



Attribute Handles are public information

Attribute Types are public information

Attribute Values can be protected

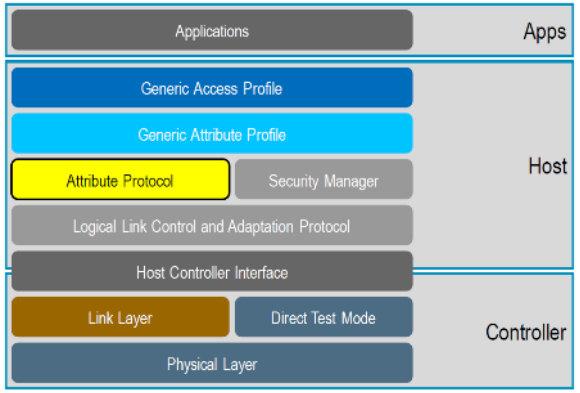
It is up to the server to not reveal any values that itconsiders are protected to a client it does not “trust” enough

Server responds by saying what is wrong – not with value

Insufficient Authentication / Authorization / Key Size

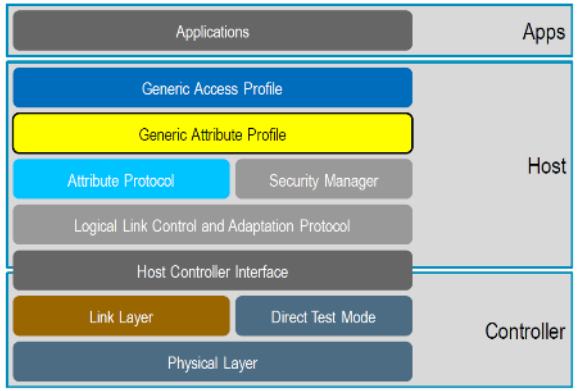
Read / Write Not Permitted

Protocol Methods



Protocol PDU Type	Sent by	Description
Request	Client	Client requests something from server – always causes a response
Response	Server	Server sends response to a request from a client
Command	Client	Client commands something to server – no response
Notification	Server	Server notifies client of new value – no confirmation
Indication	Server	Server indicates to client new value – always causes a confirmation
Confirmation	Client	Confirmation to an indication

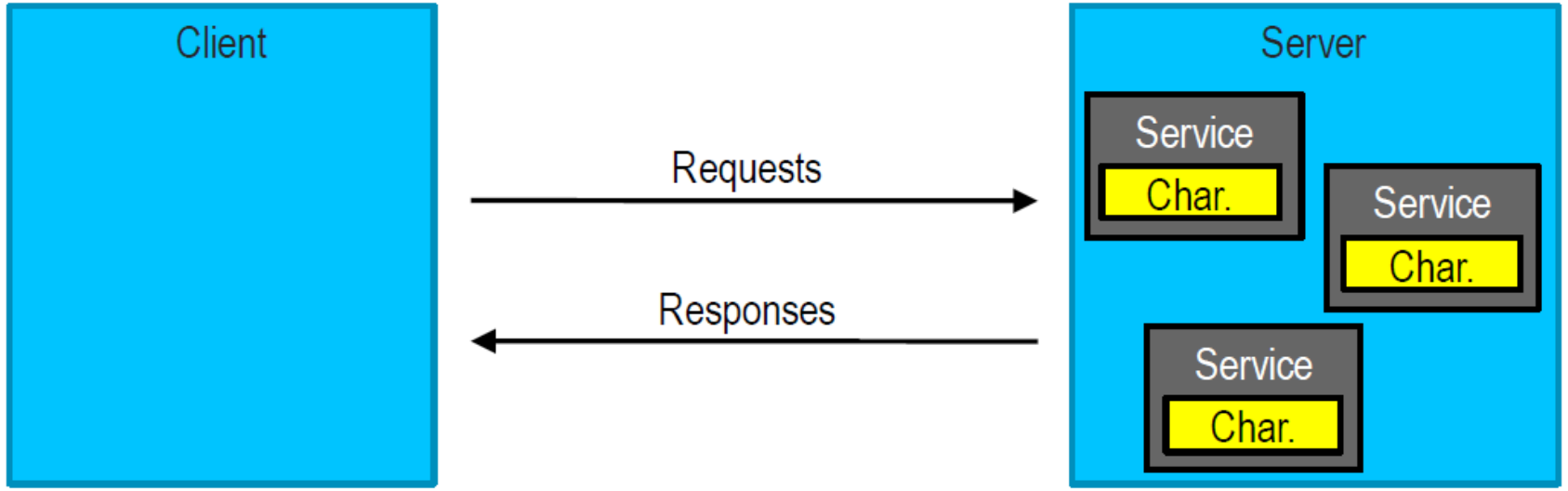
Generic Attribute Protocol (GATT)



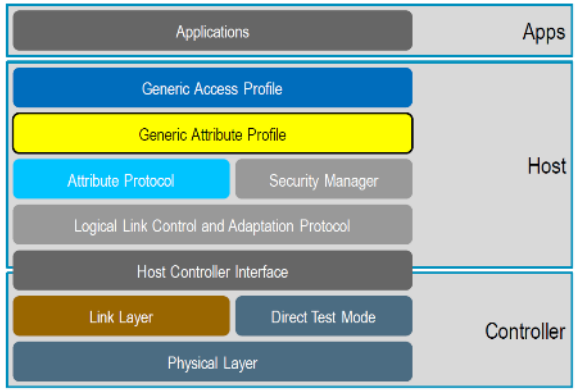
Same client server architecture as Attribute Protocol
except that data is encapsulated in “Services”
and data is exposed in “Characteristic”

Generic Attribute Protocol (GATT)
Defines concepts of:

- Service group
- Characteristic group
- Descriptors



Generic Attribute Protocol (GATT)



Generic Attribute Protocol (GATT)

Defines concepts of:

- Service group
- Characteristic group
- Descriptors

What is a Characteristic?

It's a value, with a known type, and a known format defined in a "Characteristic Specification"

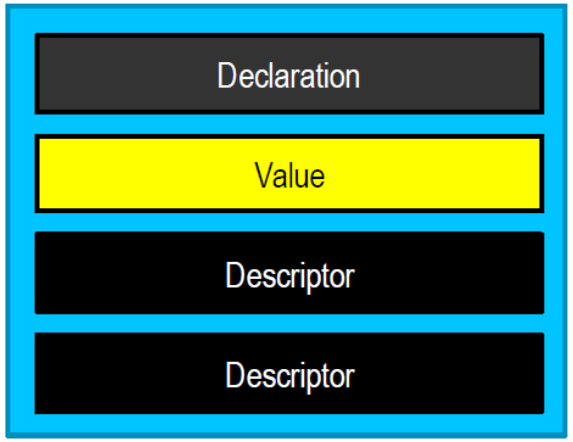
- Characteristic Declaration
- Characteristic Value
- Characteristic Descriptors

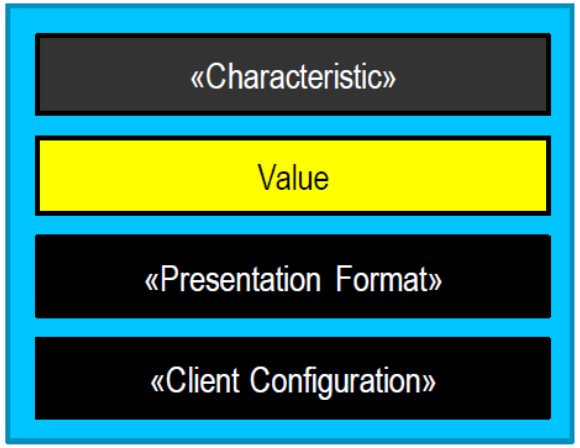
Characteristics are grouped by «Characteristic»

Value attribute is always immediately after «Characteristic» followed by descriptors

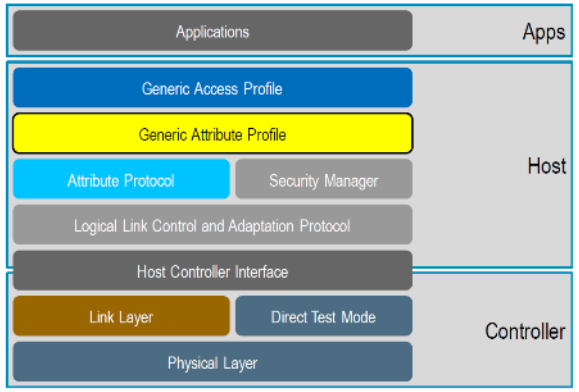
Descriptors

- additional information
- any number
- any order
- can be vendor specific





GATT: Service



Generic Attribute Protocol (GATT)

Defines concepts of:

- Service group
- Characteristic group
- Descriptors

A service is:

- defined in a “Service Specification”
- collection of characteristics
- references to other services

Service Declaration

- Includes
- Characteristics

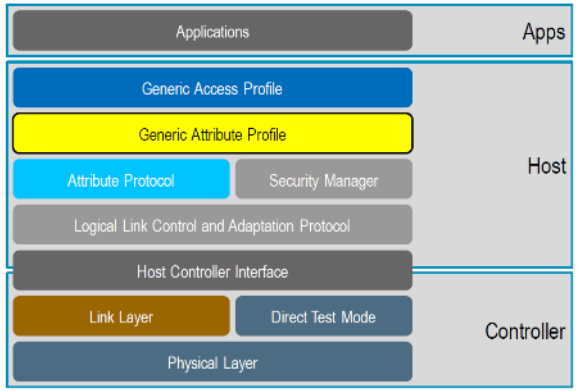
Primary Service

> *A primary service is a service that exposes primary usable functionality of this device. A primary service can be included by another service.*

Secondary Service

> *A secondary service is a service that is subservient to another secondary service or primary service. A secondary service is only relevant in the context of another service.*

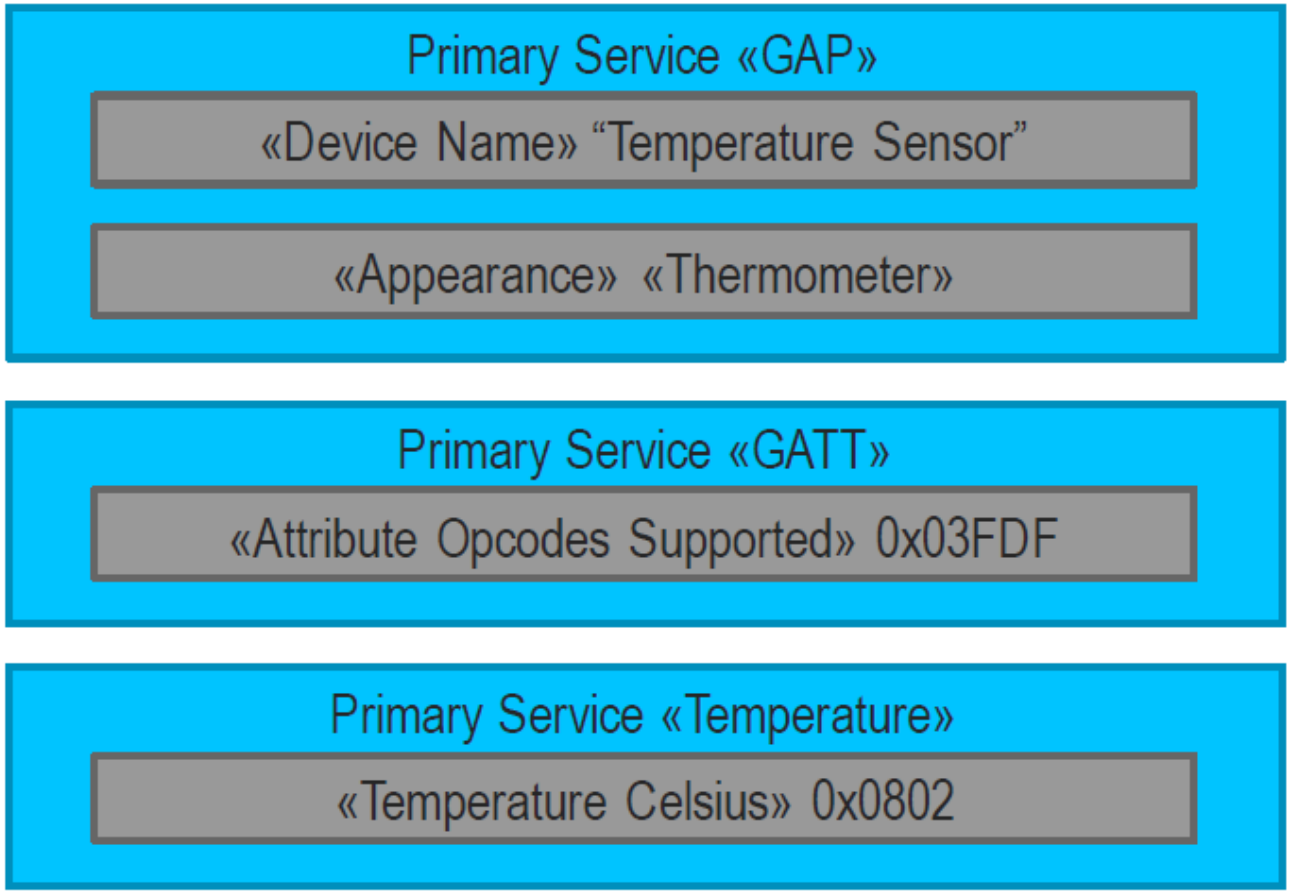
GATT: Structure



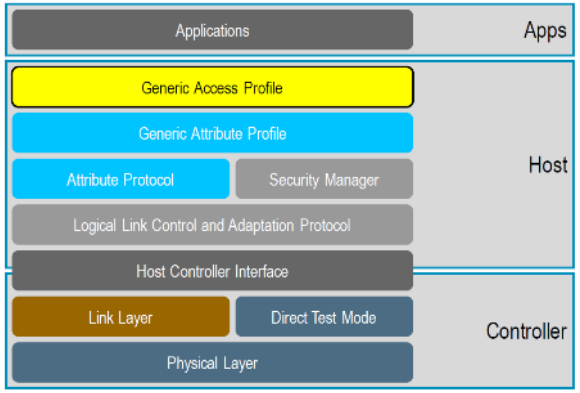
Generic Attribute Protocol (GATT)

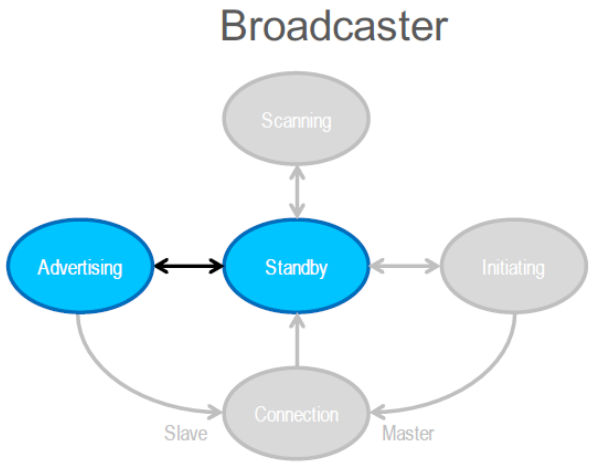
Defines concepts of:

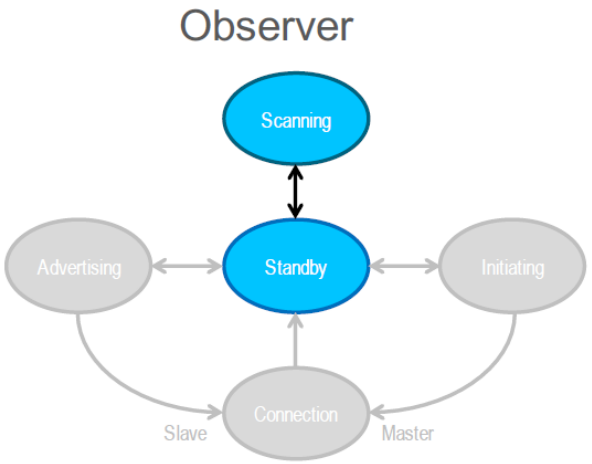
- Service group
- Characteristic group
- Descriptors

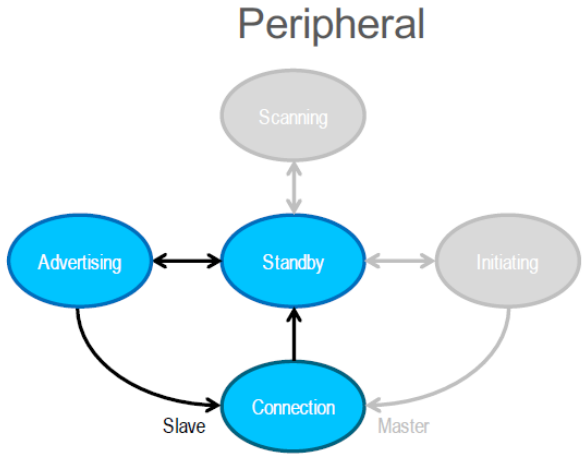


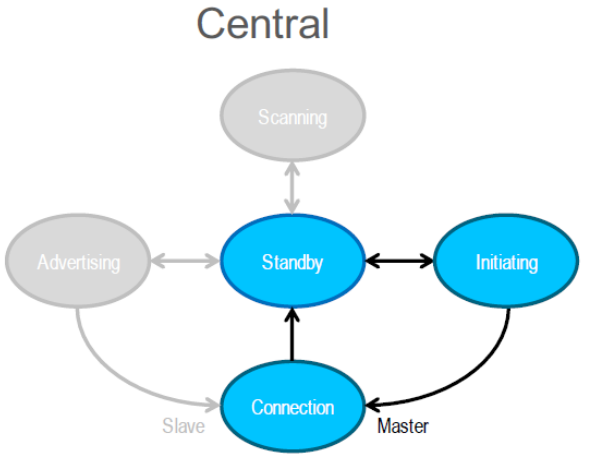
Generic Access Profile (GAP)









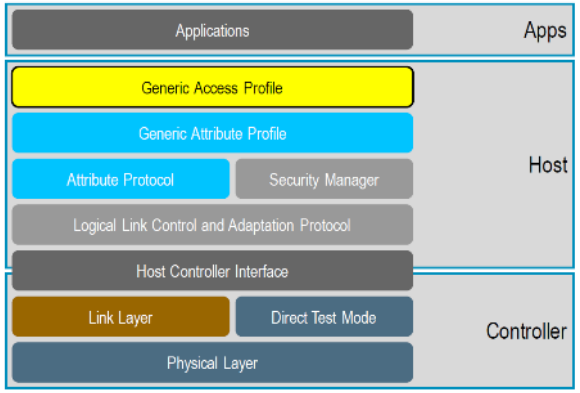


Generic Access Profile (GAP):

Profile Roles:

- Broadcaster, Observer, Pheriphera, Central
- Discoverable, Connectable, Bonding
- Privacy: Private Addresses

GAP Characteristics



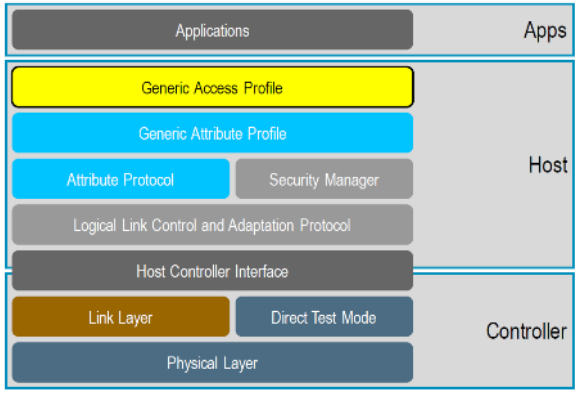
Generic Access Profile (GAP):

Profile Roles:

- Broadcaster, Observer, Pheriphera, Central
- Discoverable, Connectable, Bonding
- Privacy: Private Addresses

Characteristic	Description
«Device Name»	the local name of the device
«Appearance»	enumeration of what the device “looks like”
«Peripheral Privacy Flag»	does this peripheral support privacy – is it enabled
«Reconnection Address»	address to use when reconnecting to a private device
«Peripheral Preferred Connection Parameters»	what this peripheral would prefer the central connect with

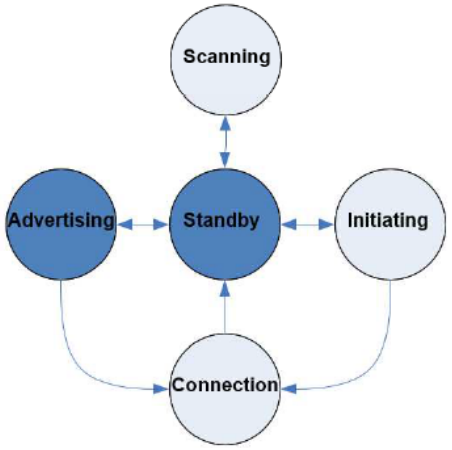
GAP Roles and Modes



- **Advertising mode**
- Connected Mode

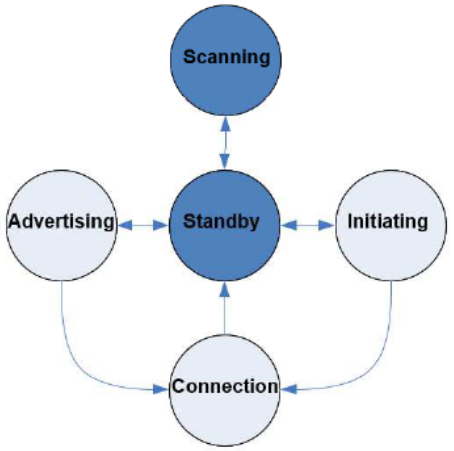
Broadcaster

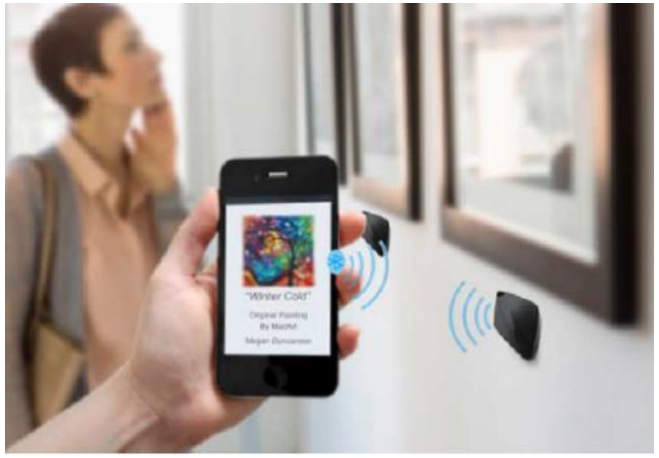
Sends advertising events
Can include characteristics and service data
Doesn't need receiver
Can be discoverable if it does have receiver



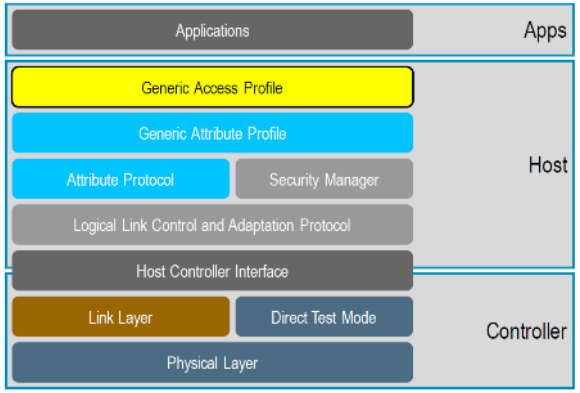
Observer

Receives advertising events
Listens for characteristics and service data
Doesn't need transmitter
Can discover devices if it does have transmitter





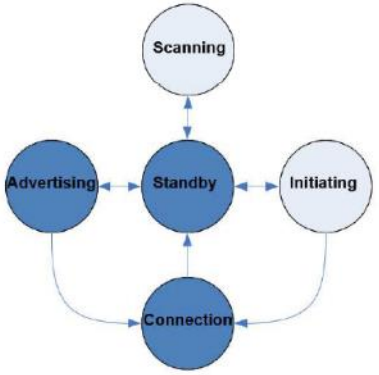
GAP Roles and Modes



- Advertising mode
- **Connected Mode**

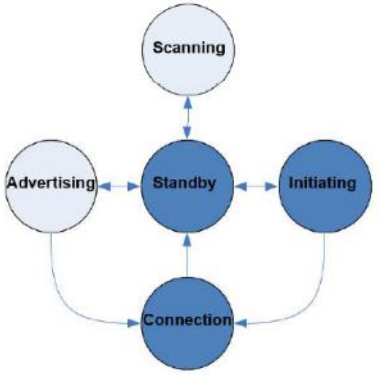
Peripheral

Has transmitter and receiver
Always slave
Connectable advertising



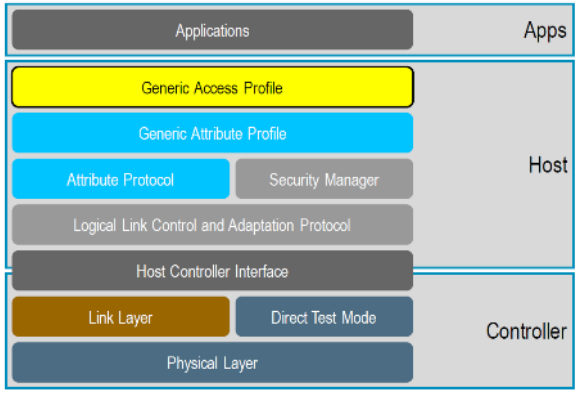
Central

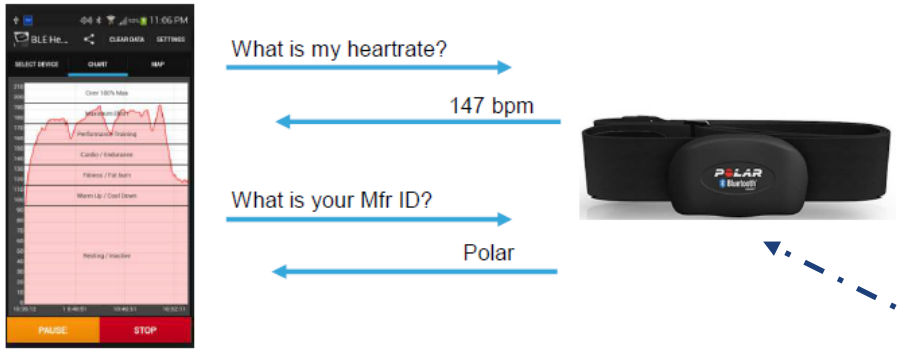
Has transmitter and receiver
Always master
Never advertises



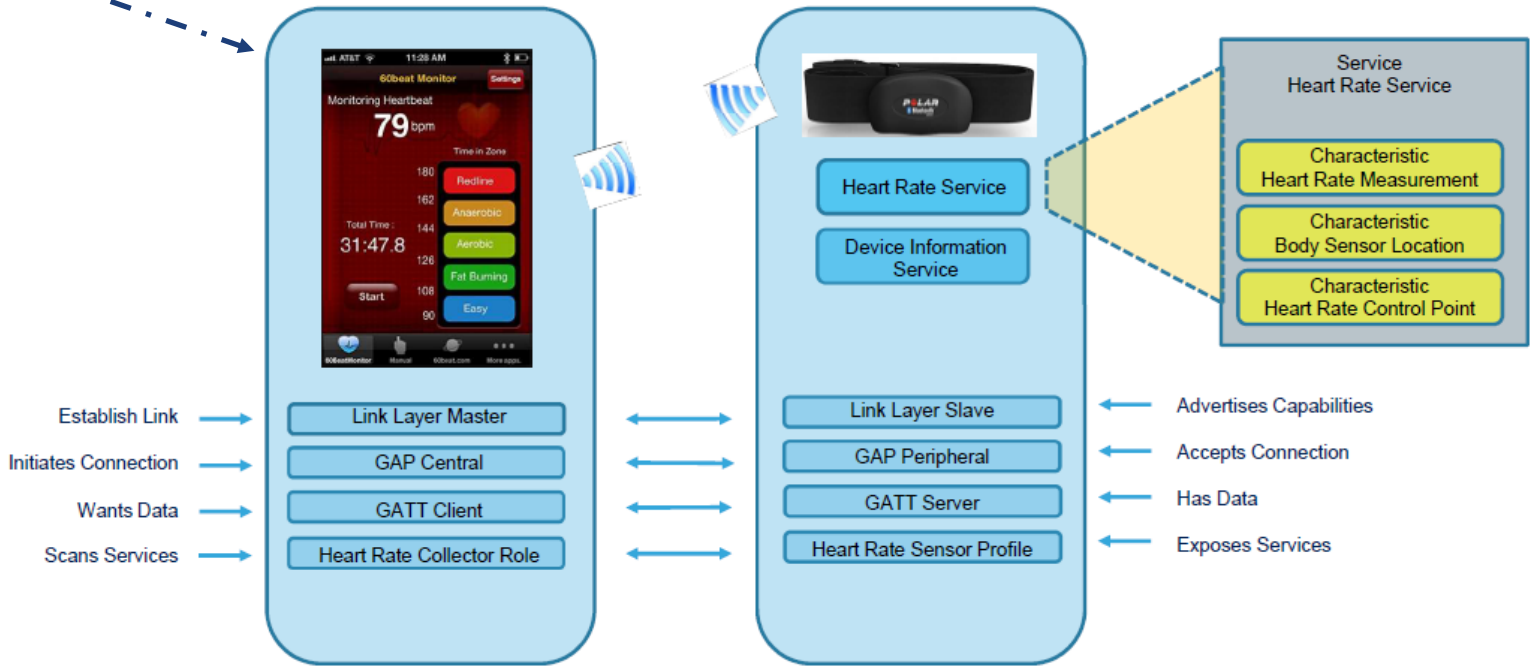
GAP Central is also a “GATT Client”
GAP Peripheral” is also a “GATT Server”

GATP: Profiles, Services, Characteristics & Descriptors





- Advertising mode
- **Connected Mode**



GAP – GATT – Attribute – HCI – Link Layer

Attribute

GATT Client

Attribute protocol (ATT)

GATT Server (Glucose profile)

Operations:
Read
Write
Indicate
Notify

Primary service Declaration UUID: 0x1800

Device Information service UUID: 0x180A

Glucose service UUID: 0x1808

Glucose Measurement Characteristic UUID: 0x2A18

Glucose Meas. Context Characteristic UUID: 0x2A34

Glucose Feature Characteristic UUID: 0x2A51

Flags (8-bit)

Sequence # (16-bit)

Carbohydrate ID (8-bit)

Carbohydrate (float)

Meal (8-bit)

Tester (nibble)

Descriptor Unit – kg UUID: 0x2702

GAP

Example! A smart BLE drill

- Battery Monitoring Service (BAS)
- Alert Notification Service (ANS)
- Elapsed motor use in minutes
- Unlock the drill via smartphone password
  - Add standard Services & Characteristics (16-bit UUID's from Bluetooth SIG)
  - Create custom services (128-bit UUID's)

Example! A smart BLE drill

- Standard Services & Characteristics (16-bit UUID's)
  - Battery service (BAS) UUID: 0x180F
    - *Battery Level Characteristic*: 0x2A19
  - Alert Notification Service UUID: 0x1811
    - *Alert Notification Control Point Characteristic:* 0x2A44
    - *Unread Alert Status Characteristic :* 0x2A45
    - *New Alert Characteristic :* 0x2A46
    - *Supported New Alert Category Characteristic:* 0x2A47
    - *Supported Unread Alert Category Characteristic :* 0x2A48
- Create custom services (128-bit UUID's)
  - **10c17863-9471-4427-8d66-82579bf9161a** (Motor run time service)
    - **5567fa77-721f-4e1a-9875-7ae95ead642d** xxx Characteristic
    - **3d78d6f3-7d34-4f89-a14d-ed3cac297438** xxx Characteristic
  - **0226b0db-d9a6-49c8-bce1-fccd3a40e6e2** (Unlock service)
    - **997e28a5-f05e-4027-89c7-e84ce4ce67ec** xxx Characteristic
    - **b3b7d2a1-4eeb-4a39-85ef-7ddd7b1e4abf** xxx Characteristic

Name: Battery Service

Type: org.bluetooth.service.battery_service

Assigned Number: 0x180F

Name: Battery Level

Type: org.bluetooth.characteristic.battery_level

Assigned Number: 0x2A19

Name: Alert Notification Service

Type: org.bluetooth.service.alert_notification

Assigned Number: 0x1811

Name: Alert Notification Control Point

Type: org.bluetooth.characteristic.alert_notification_control_point

Assigned Number: 0x2A44

Name: Supported New Alert Category

Type: org.bluetooth.characteristic.supported_new_alert_category

Assigned Number: 0x2A47

fe.augmented

STM32 WB

We can generate a random UUID for it at https://www.uuidgenerator.net/

Dig Deeper

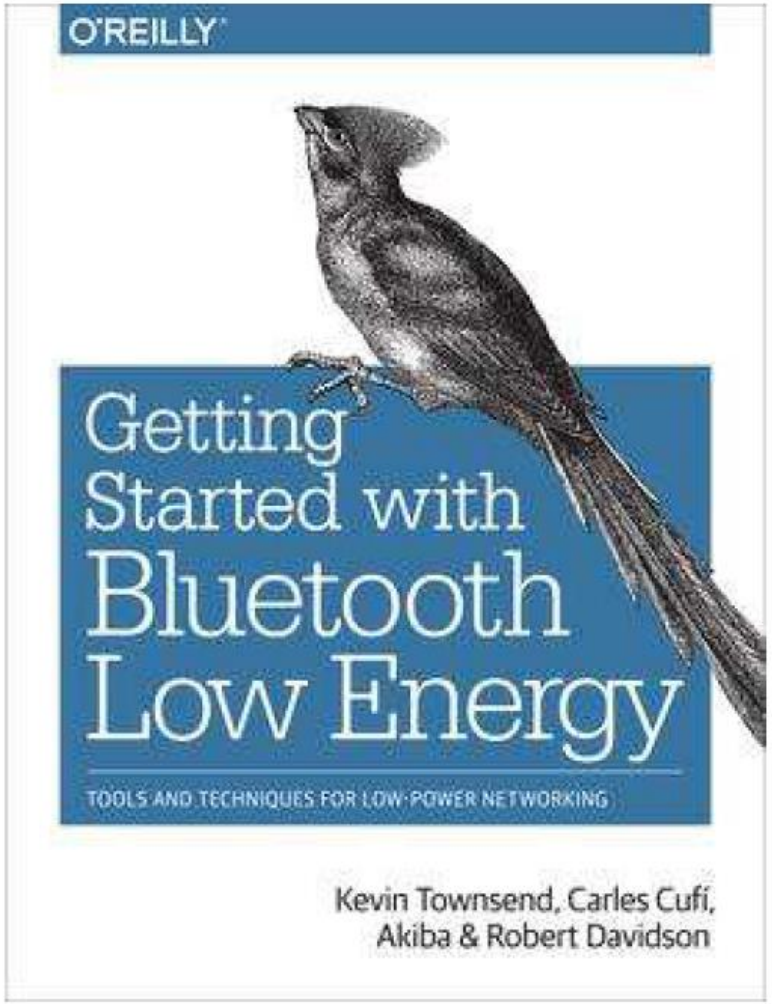
Bluetooth https://www.bluetooth.com/

Core Specifications

The *Bluetooth*® Core Specification defines the technology building blocks that developers use to create the interoperable devices that make up the thriving Bluetooth ecosystem. The Bluetooth specification is overseen by the Bluetooth Special Interest Group (SIG) and is regularly updated and enhanced by Bluetooth SIG Working Groups to meet evolving technology and market needs.

Specification		Version	Status	Adoption Date
CS	Core Specification	5.0	Active	06 Dec 2016
CSS	Core Specification Supplement	7	Active	06 Dec 2016
CSA	Core Specification Addendum	6	Active	12 Jul 2017

36$ on Amazon



Dig Deeper

https://www.bluetooth.com/specifications/gatt

- Working Groups
- Core Specifications
- Mesh Networking Specifications
- Traditional Profile Specifications
- Protocol Specifications
- **GATT Specifications**
  - GATT Overview
  - GATT Characteristics
  - GATT Declarations
  - GATT Descriptors
  - GATT Services
  - Mesh GATT Services XML
  - Available Schemas
- Errata Service Releases
- Qualification Test Requirements
- Assigned Numbers

GATT Specifications

Generic Attributes (GATT) services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device.

A *GATT profile* describes a use case, roles, and general behaviors based on the GATT functionality, enabling extensive innovation while maintaining full interoperability with other *Bluetooth*® devices.

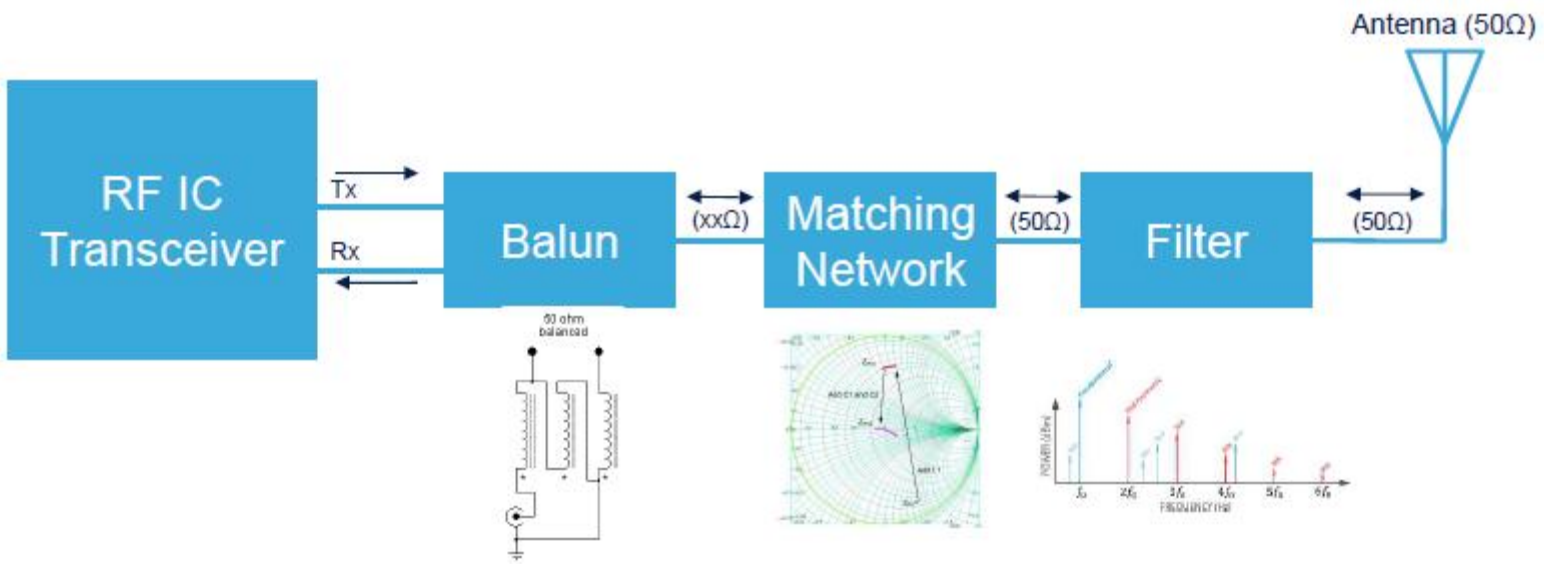
The documents in the "Informative document showing changes" column are provided as a courtesy to help readers identify changes between two versions of a Bluetooth specification. When implementing specifications, use the adopted versions in the "Adopted Version" column.

More about GATT

Profile Specification		Version	Status	Adoption Date	Informative document showing changes
ANP	Alert Notification Profile	1.0	Active	13 Sep 2011	N/A
ANS	Alert Notification Service	1.0	Active	13 Sep 2011	N/A
AIOP	Automation IO Profile	1.0	Active	14 Jul 2015	N/A
AIOS	Automation IO Service	1.0	Active	14 Jul 2015	N/A
BAS	Battery Service	1.0	Active	27 Dec 2011	N/A
BCS	Body Composition Service	1.0	Active	21 Oct 2014	N/A

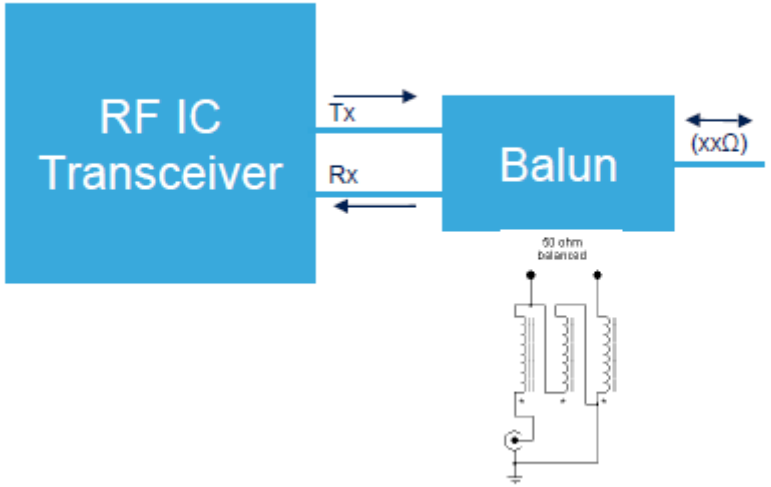
RF System – Front-End

- **Balun** – Combine TX and RX signals
- **Matching Network** – 50 Ω impedance transformation
- **Harmonic Filter** – Reduce out-of-band harmonics
- **Chip Antenna** – Comparable performance with lambda antennas



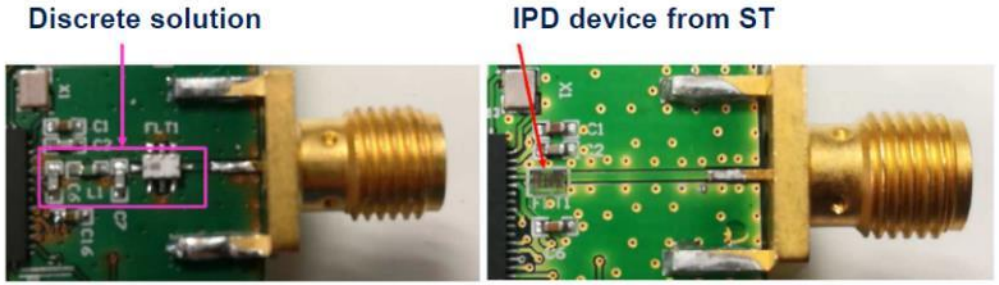
RF System – Front-End

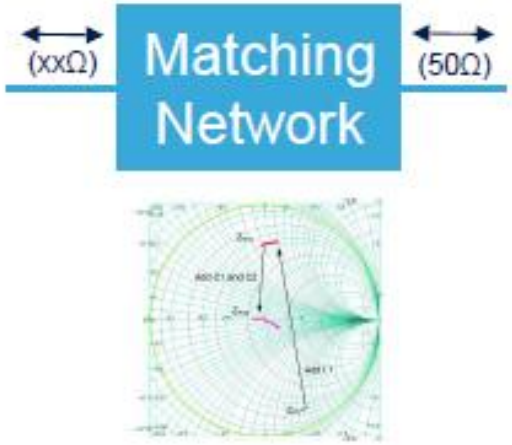
- **Balun** – Combine TX and RX signals
- **Matching Network** – 50 Ω impedance transformation
- **Harmonic Filter** – Reduce out-of-band harmonics
- **Chip Antenna** – Comparable performance with lambda antennas

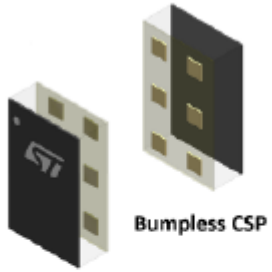


RF System – Front-End

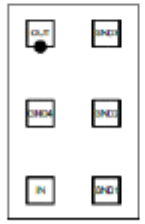
- **Balun** – Combine TX and RX signals
- **Matching Network** – 50 Ω impedance transformation
- **Harmonic Filter** – Reduce out-of-band harmonics
- **Chip Antenna** – Comparable performance with lambda antennas







Top view (pads down)



Features

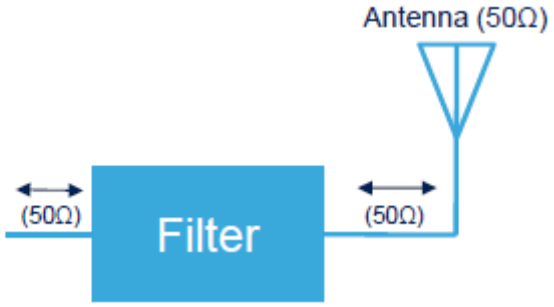
- Integrated impedance matching to STM32WB55Cx
- LGA footprint compatible
- 50 Ω nominal impedance on antenna side
- Deep rejection harmonics filter
- Low insertion loss
- Small footprint
- Low thickness ≤ 450 µm
- High RF performance
- RF BOM and area reduction
- ECOPACK®2 compliant

Applications

- Bluetooth 5
- OpenThread
- Zigbee®
- IEEE 802.15.4
- Optimized for STM32WB55Cx and STM32WB55Rx

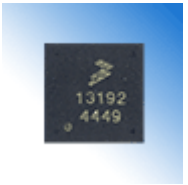
RF System – Front-End

- **Balun** – Combine TX and RX signals
- **Matching Network** – 50 Ω impedance transformation
- **Harmonic Filter** – Reduce out-of-band harmonics
- **Chip Antenna** – Comparable performance with lambda antennas

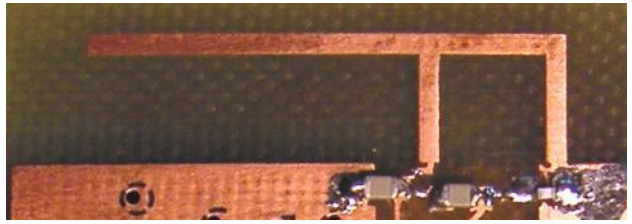


RF System – Front-End

- **Transceiver**
  - Communication with other devices
  - Consumption (10 - 100mA)
  - Start-up time and available bandwidth



- **Antenna**
  - Chip antenna
  - PCB antenna
  - External antenna



Transmit power and current consumption

TX PA current typically starts dominating the total current consumption for output power levels above 0 dBm

- Most BLE applications run at 0 dBm output power
- Average current consumption at 0 dBm is only 6 mA
- **Output up to 5 dBm while keeping the current consumption below 10 mA**
- Allows the application to run off common coin cell batteries