

Mobile Computing Bluetooth Low Energy Personal Area Networks

CC BY-SA, T. Amberg, FHNW

Slides: tmb.gr/mc-ble

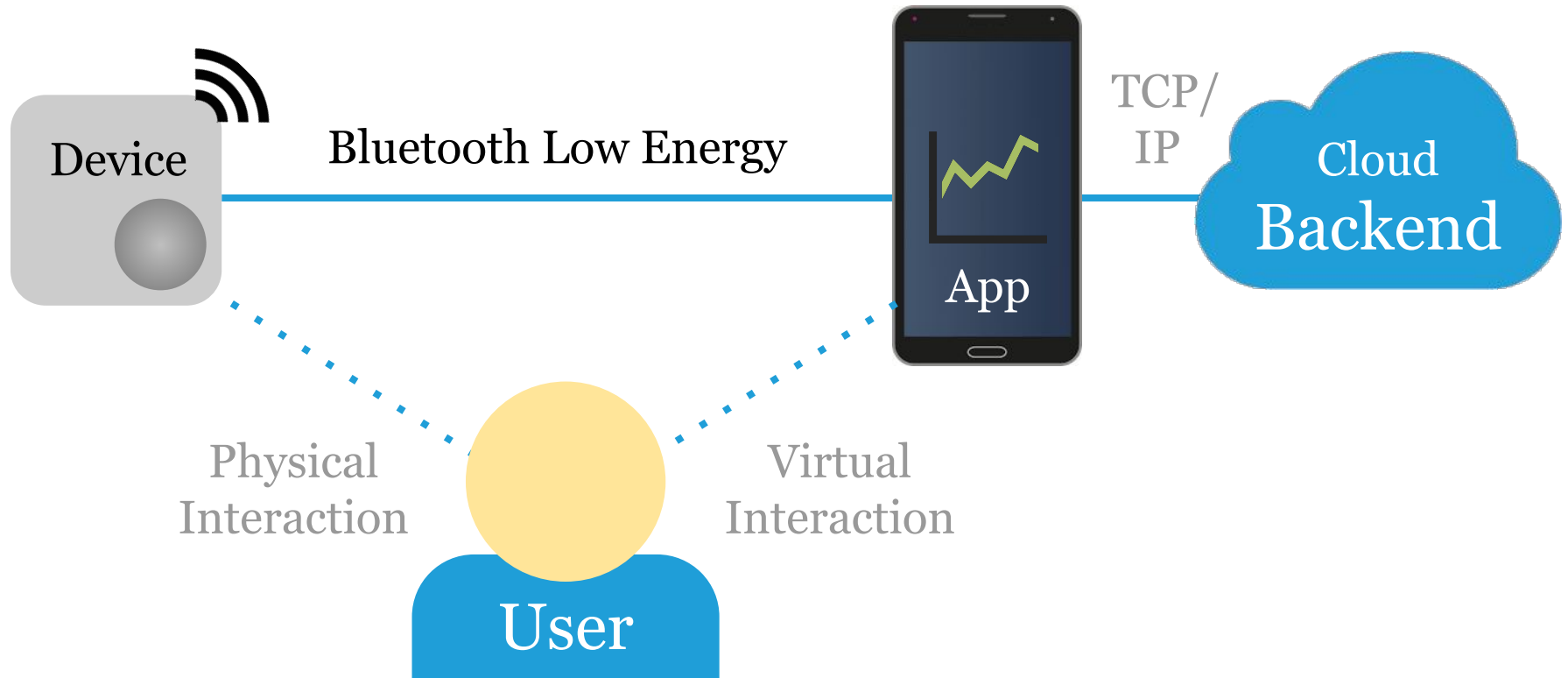
Overview

These slides introduce *Bluetooth Low Energy*.

Which are the roles of involved parties.

How a BLE service is structured.

Reference model



Bluetooth Low Energy (BLE)

BLE is a power-efficient Bluetooth variant (since 4.0).

BLE is well suited for small, battery powered devices.

BLE uses less energy than Wi-Fi, way less than 3/4G.

Both, classic Bluetooth and BLE, use 2.4 GHz radio.

The **standard** is maintained by the **Bluetooth SIG**.

How BLE works

Peripherals advertise the data they have, over the air.

Centrals scan for nearby peripherals to discover them.

The central connects to a peripheral and uses its data.

Data is structured into *services* and *characteristics*.

BLE protocol stack

Application — application specific code and formats

BLE library — thin, language-specific wrapper library

GATT — services & characteristics | GAP — discovery

ATT — attribute transport | SMP — security manager

L2CAP — logical link control and adaptation protocol

Link layer — exposed via the host controller interface

Physical layer — dealing with actual radio signals

Generic Access Profile (GAP)

GAP defines the following roles, communication types:

Broadcaster and *observer* (connectionless, one-way).

Peripheral and *central* (bidirectional connection).

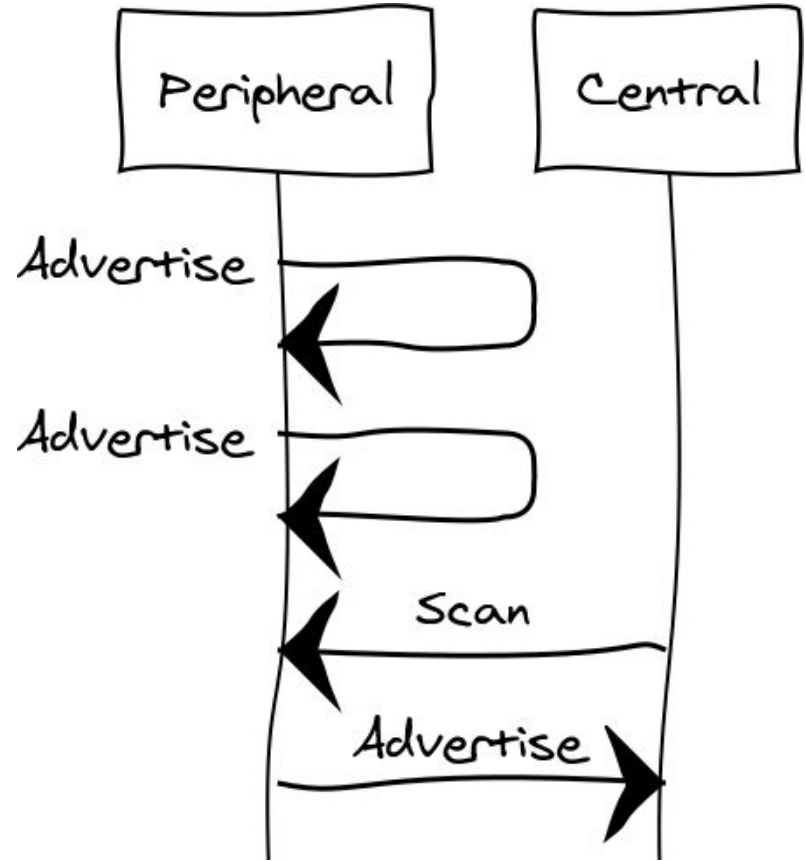
Each device supports one or more of these roles.

We start with peripheral and central roles.

Advertising

A peripheral *advertises* its services by broadcast, in a regular **interval**.

A central *scans* for all or a subset of services and gets device addresses and, if it's been sent, advertised data.



Attribute Transport (ATT)

ATT allows a *client* to access attributes on a *server*.

An *attribute* has a UUID, a handle and permissions.

A *UUID* is a 32/128-bit universally unique identifier.

An *attribute handle* is a server-assigned, 16-bit ID.

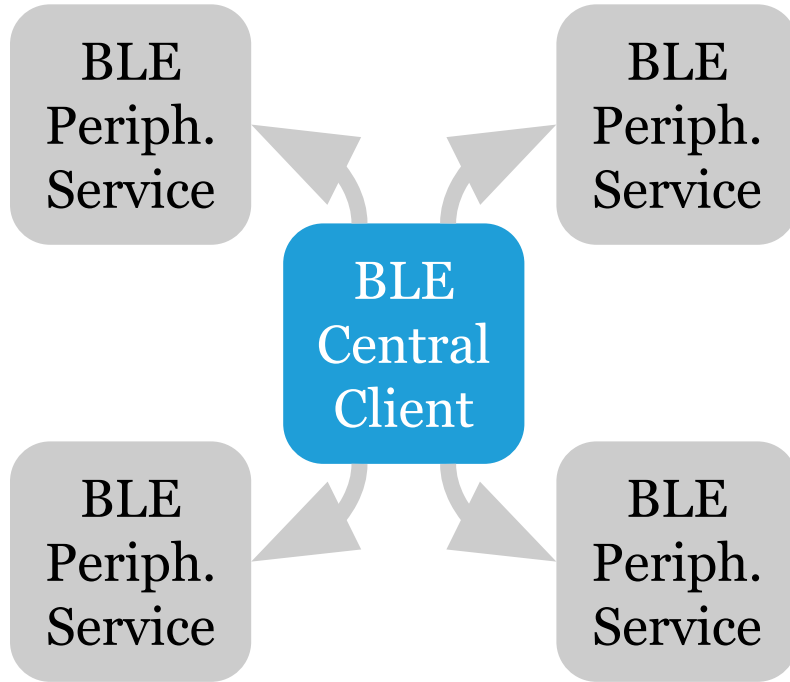
See [Bluetooth spec v5.1](#), Part F, p. 2288.

Generic Attribute Profile (GATT)

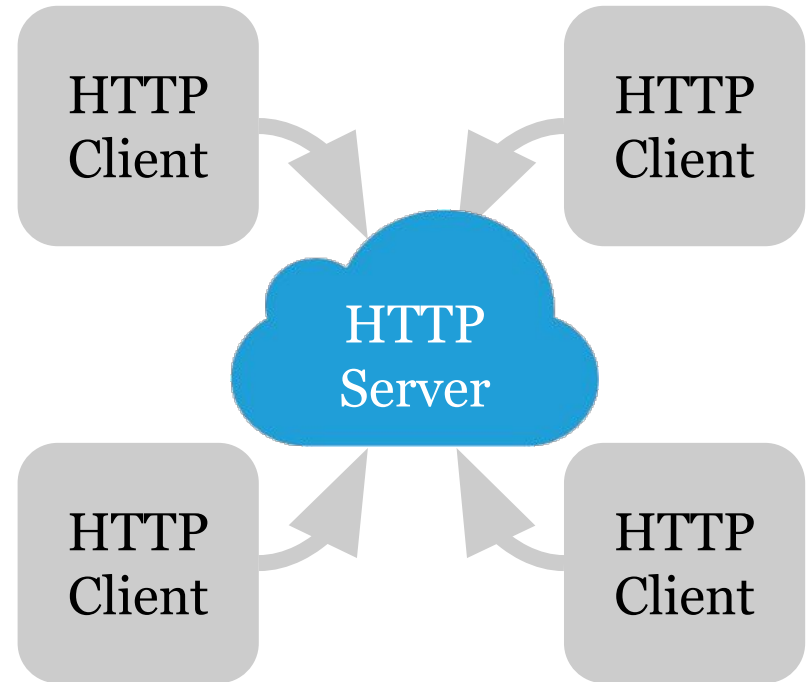
GATT is a simple application level protocol for BLE.
It's connection-based, with a *client* and a *server* role.
This enables a BLE device to provide a RESTful API.
A "GATT API", or *profile*, is a collection of *services*.

Usually, the peripheral acts as a server.

BLE



HTTP



Services

A **GATT service** is a collection of characteristics.

Services encapsulate the behavior of part of a device.

In addition, such a service can refer to other services.

There are **standard** and custom services and profiles.

E.g. the **Battery Service** or the **Heart Rate Service**.

Characteristics

A **GATT characteristic** has a value and descriptors.

A *value* encodes data "bits" that form a logical unit.

Descriptors are defined attributes of a characteristic.

Supported procedures: read, write and notifications.

E.g. a **Battery Level** or a **Heart Rate Measurement**.

Descriptors

A **GATT descriptor** describes a characteristic value.

E.g. **Presentation Format** or **Valid Range** descriptor.

Descriptors also allow to configure characteristics.

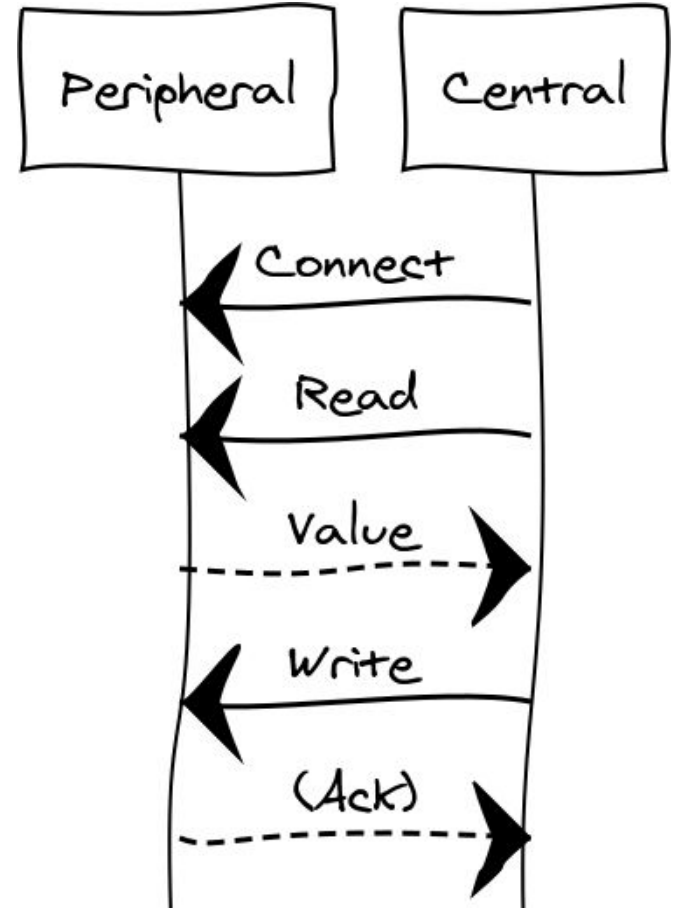
E.g. **Client Characteristic Configuration** descriptor allows a client to enable or disable notifications.

Read and write

Connect = the central connects to a peripherals BLE address.

Read = value of a characteristic or its descriptors is returned.

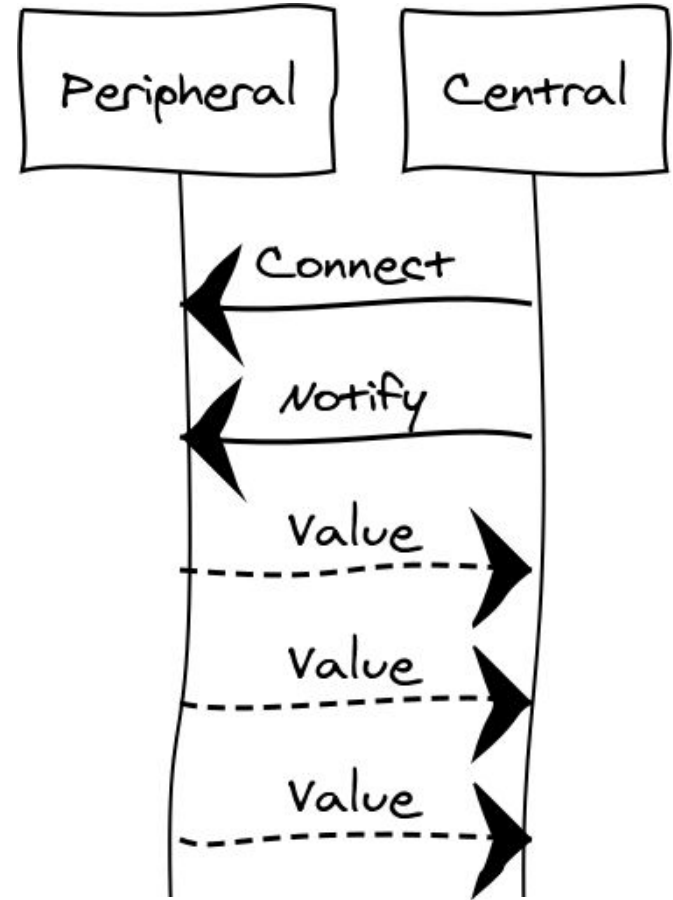
Write = characteristic value, or characteristic descriptor value is set, with/without response.



Notifications

Notify = *Client Characteristic Configuration* descriptor of a characteristic, UUID 0x2902, is set to 0x0000 using *Write*.

Value = *A Handle Value Notification* is sent if value changes.



See [Bluetooth spec v5.1](#), p. 2360 and p. 2389.

BLE explorer apps

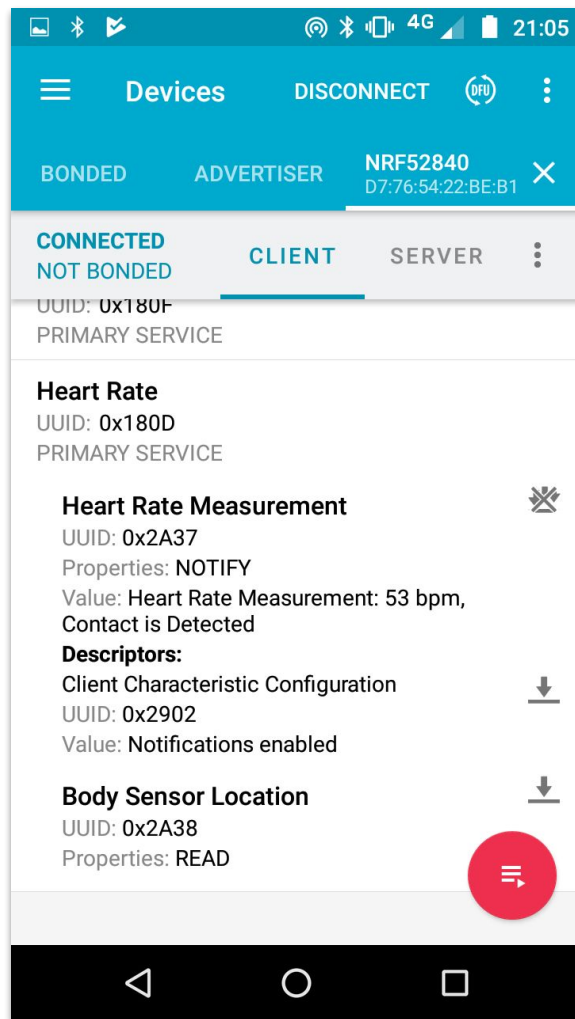
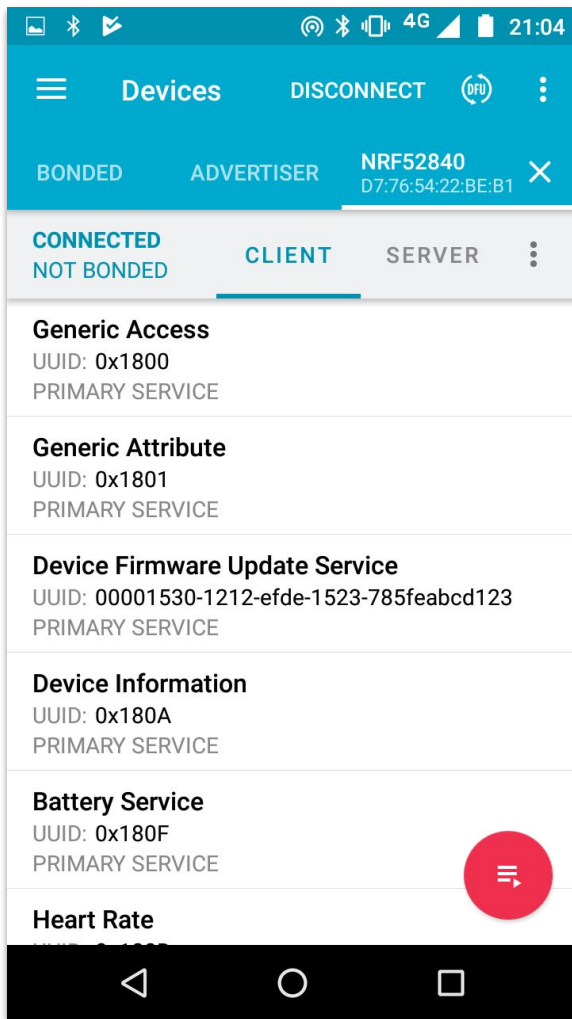
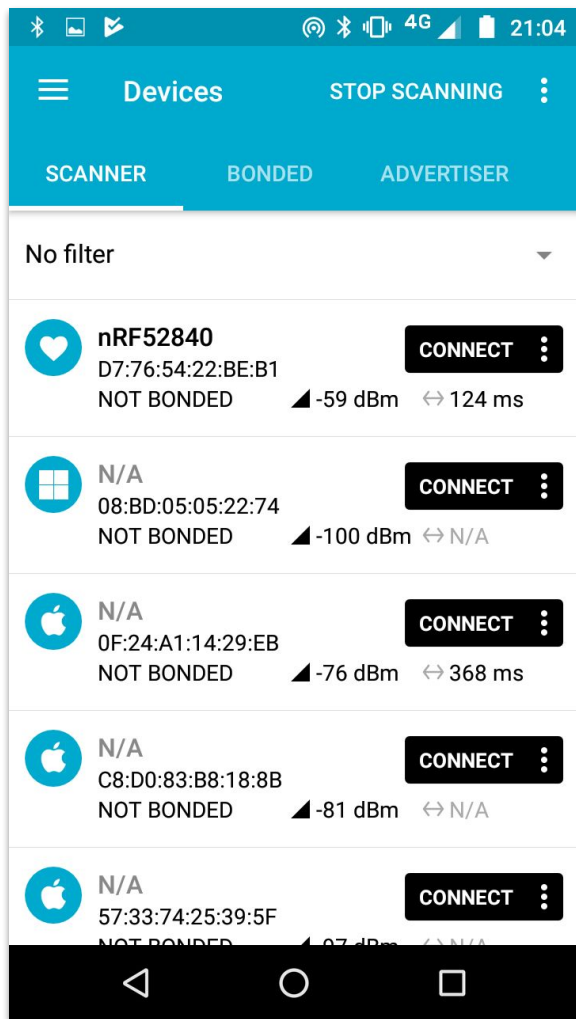
For debugging, use any generic BLE explorer app:

Find **BLE explorer apps** on the Google Play Store.

Search for "BLE explorer" in the iOS App Store.

Smartphones can act as central or peripheral.

Exploring is a great way to learn about BLE.



Heart Rate Service

This service is intended for fitness heart rate sensors:

Heart Rate Service UUID (16-bit): 0x**180D**

This service includes the following characteristics:

Heart Rate Measurement UUID: 0x**2A37** [N]

Body Sensor Location UUID: 0x**2A38** [R]

Heart Rate Control Point UUID: 0x**2A39** [W]*

*See also **Heart Rate Service** specification.

Nordic UART Service

This service provides a serial connection over BLE:

Nordic UART Service custom (128-bit) UUID:
6E400001-B5A3-F393-E0A9-E50E24DCCA9E

This service includes the following characteristics:

RX (device receives data) UUID: 0x**0002** [W]

TX (device transmits data) UUID: 0x**0003** [N]

This service is becoming a de facto standard.

Beacons

Beacons, e.g. [Apple iBeacon](#) are *broadcaster* devices.

Any *observer* can read the data which they advertise.

Lookup of "what a beacon means" requires an app.

Except for [Physical Web](#) / [Eddystone](#) beacons.

These contain an URL to be used right away.

Security

BLE has **security mechanisms** for pairing and more.

Pairing: exchanging identity and keys to set up trust.

Device chooses *Just Works*, *Passkey Entry* or **OOB**.

Or numeric comparison and **ECDH** for key exchange.

Some apps add encryption* on the application layer.

*See, e.g. HomeKit in **iOS Security Guide** (p. 29).

Summary

BLE provides low power, personal area connectivity.

A BLE central scans for peripherals, who advertise.

Each BLE peripheral provides one or more services.

Services allow to read/write characteristic values.

Descriptors allow to configure notifications.

Feedback or questions?

Join us on [MSE TSM MobCom](#) in MS Teams

Or email thomas.amberg@fhnw.ch

Thanks for your time.