Introduction to BLE in iOS

Elliot Sinyor

CocoaHeads Montréal, May 10, 2016

Contents

- Intro
- Basic Concepts
- Example of Discovery + Connection
- Background Operation / Security
- Demo

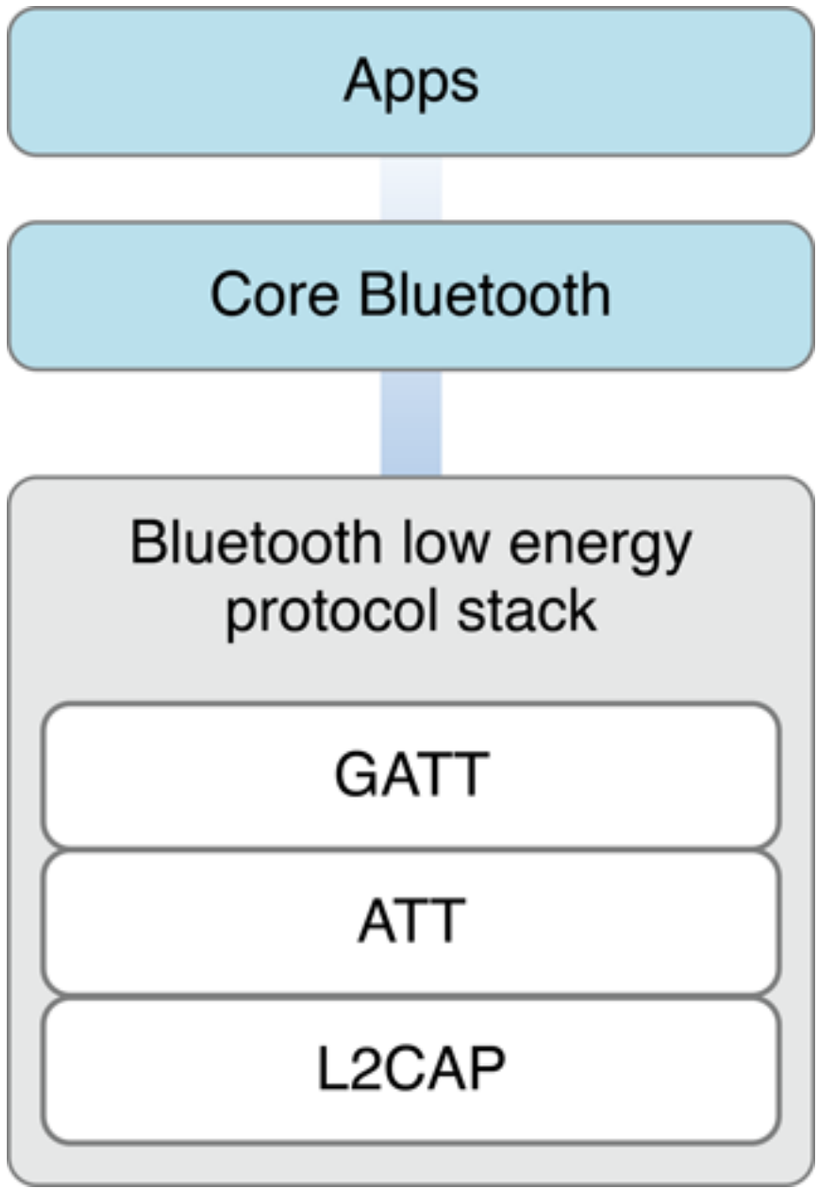
Why use it?

- **Low Energy**
  - Generally matters more for small hardware devices
    - eg: Wearables, IoT sensors, Controlling devices
      - Battery life can be measured in weeks/months, not hours
- Relatively Accessible
  - Other than wifi, the only (practical) way to communicate with 3rd party hardware

Classic vs. LE in numbers

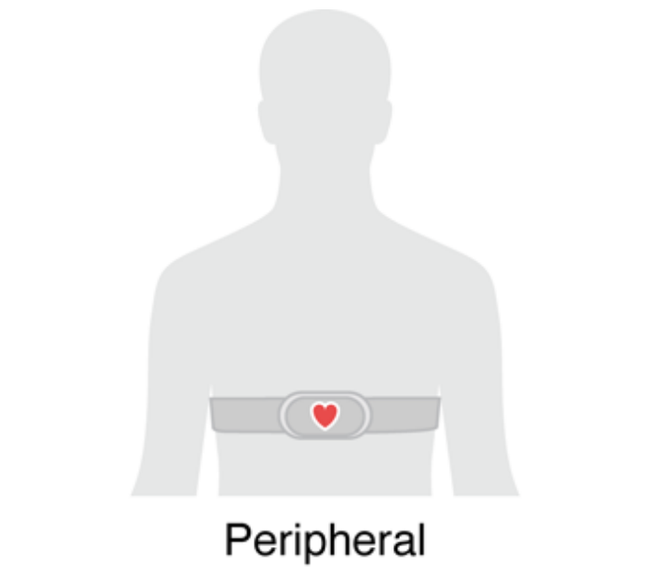
	Bluetooth Classic (up to v3.x)	Bluetooth Low Energy (4.0 +)
Latency (connected):	30ms	6ms
Throughput:	0.7 - 2.1 Mbit/s	0.27 Mbit/s
Range:	100m	> 100m
Avg Power Consumption:	~1W	0.01 W - 0.5 W

iOS Stack


Apps
Core Bluetooth
Bluetooth low energy protocol stack
GATT
ATT
L2CAP

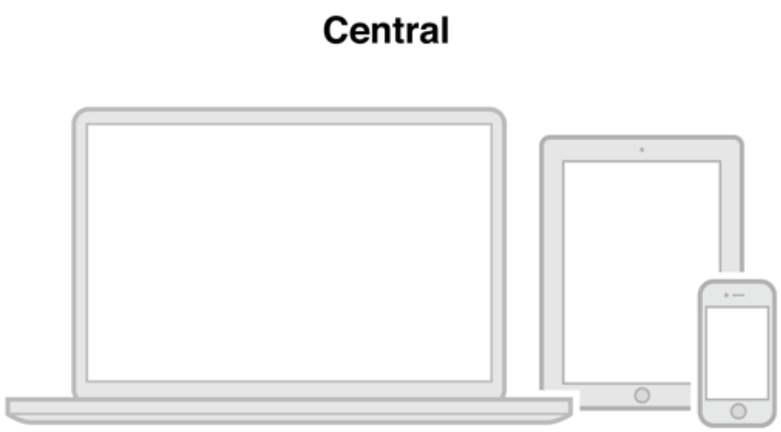
Central / Peripheral

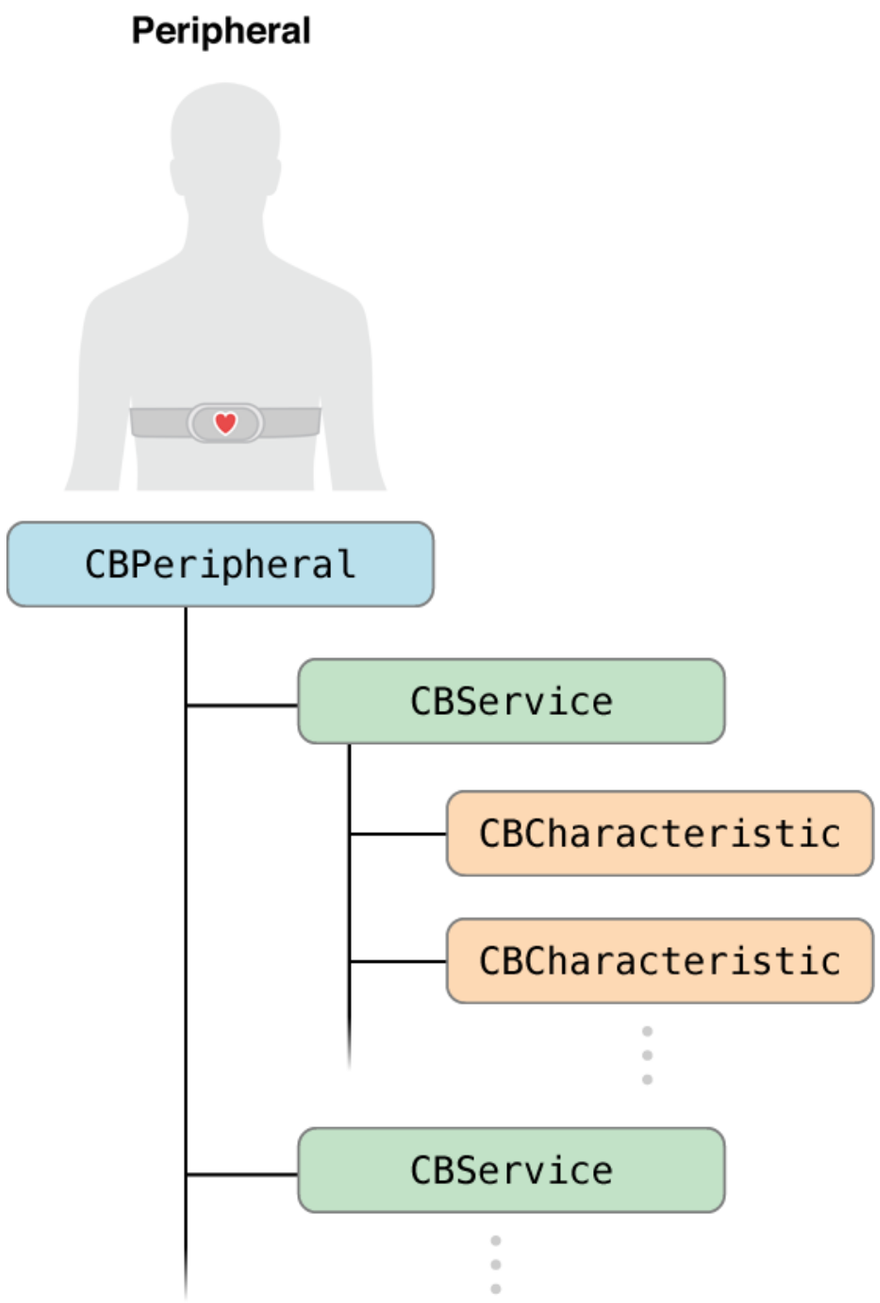
Services / Characteristics



- Both Services and Characteristics are represented by UUID
  - eg: 07286EEB-5985-4B7E-9F15-C3CA9FBE519D

Central Side Objects



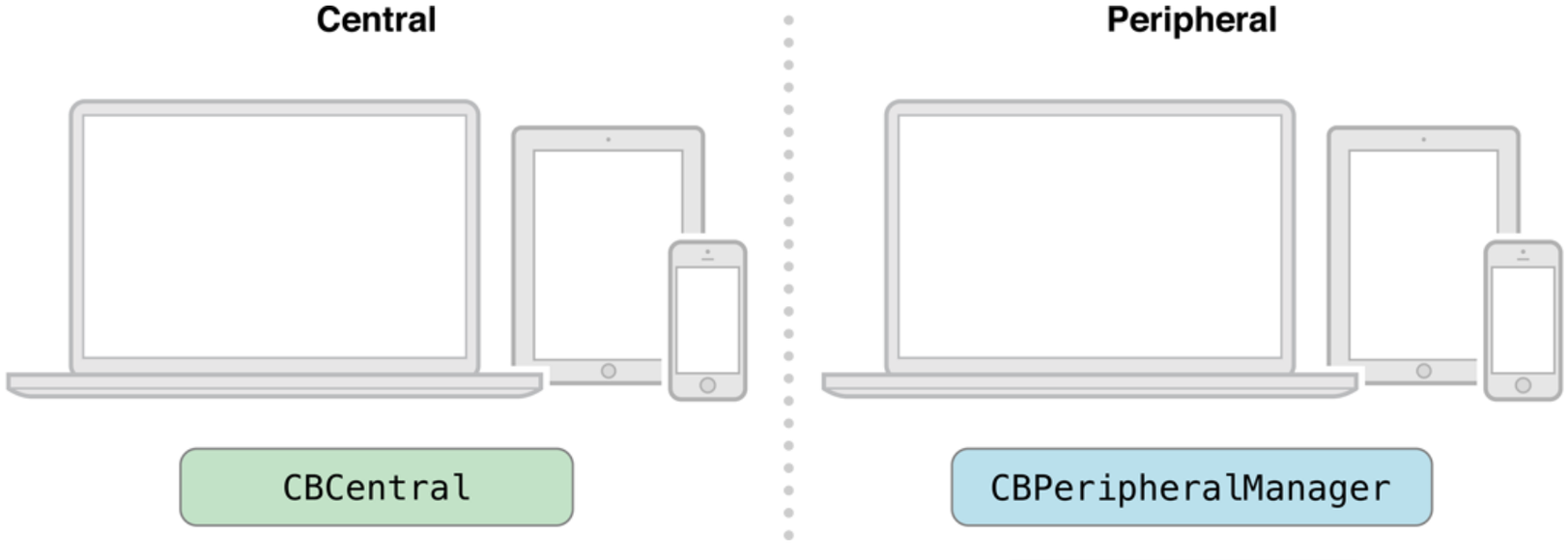


CBCentralManager

- Provides methods + defines delegate methods for operating the central

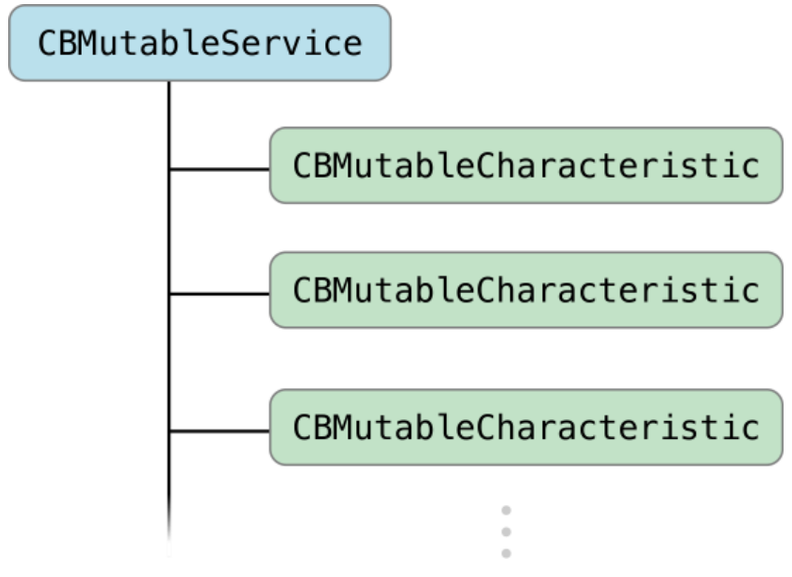
eg: scanning, connecting, reading

Peripheral Side Objects



CBPeripheralManager:

Provides methods + defines delegate methods to control Peripheral



eg: advertising, setting up services, providing data on request

Characteristic Properties (mainly):

Read, Write, Notify, Indicate

Advertising

- Central is given Advertising Dictionary + RSSI information
- When starting a scan, you can specify whether you want to get a callback for each individual advertisement

Discovery Process

Central

Peripheral

```
peripheralManager.addService(…)

peripheralManager.startAdvertising(…)
```

```
centralManager.scanForPeripheralsWithServices(<list of UUIDS>)

func centralManager(central: CBCentralManager, didDiscoverPeripheral peripheral:CBPeripheral,
advertisementData: [String : AnyObject] …)

    centralManager.connectPeripheral(…)
```

```
func centralManager(central: CBCentralManager, didConnectPeripheral peripheral: CBPeripheral)

    peripheral.discoverServices(…)

func peripheral(peripheral: CBPeripheral, didDiscoverServices error: NSError?)

    peripheral.discoverCharacteristics([charactertisticUUID], forService: service)

func peripheral(peripheral: CBPeripheral, didDiscoverCharacteristicsForService service:
CBService, error: NSError?)

    myCharacteristic = characteristic

    myPeripheral.writeValue(message.rawData, forCharacteristic:
    myCharacteristic, type: CBCharacteristicWriteType.WithoutResponse)
```

Reading / Writing

- Central initiates Writes + Reads to/from peripheral's characteristics

- What if we want the peripheral to "push" something to the central?

- Set characteristic's property to "Notify" or "Indicate"

- Have the central subscribe to that property

Background Activity

- App can act as central or peripheral in the background
    - In Info.plist set UIBackgroundModes key to either bluetooth-central or bluetooth-peripheral
- Peripheral can respond to read/write requests in background
    - Advertising rate is slower, so is response time
- Central can scan for peripherals in the background. Can also request a connection to a known peripheral
    - iOS will handle reconnection and wake up your app in the background to perform tasks
    - Slower scan times, no scan options
- Advanced: Look into “State Preservation + Restoration” + iBeacons*

Security

- v4.x uses128-bit AES encryption (CCM)
- PIN-based pairing for device with display and/or keyboard
- “Just Works” for devices with no keyboard or display
    - Both can be compromised by eavesdropper getting packet at key exchange during initial connection
    - Use Application-level encryption!
- v4.2 now uses Elliptical Curve Diffie-Hellman for exchanging keys
- Retroactively enabled on iPhone6, iPad Air2 and later devices
- More secure?

Demo

Questions?

elliot@elliotsinyor.com

https://github.com/unseenmachines