

CS123 - Communication

Programming Your Personal Robot

Kyong-Sok “KC” Chang, David Zhu
Fall 2015-16

Course Description

An introduction to the programming of a sensor-rich personal robot. This course extends programming from the virtual environment into the physical world, which presents unique challenges. Focus is on three areas of intellectual discourse that are fundamental to the programming of physical devices: communication with the devices; programming of event driven behaviors; and reasoning with uncertainty. The concepts introduced will be put into practical use through a series of class projects centered around programming your personal robot. This course also serves as a good introduction to Experimental Robotics by exposing students to basic concepts and techniques that are relevant for real world robot programming.

Course Structure

- Lectures
 - Cover basic concepts
- Readings
 - Provide background and deeper knowledge on relevant topics
- Projects
 - Hands-on experience, learning by doing
- “Freestyle”
 - Going beyond class material on your own. Hamster is an open platform

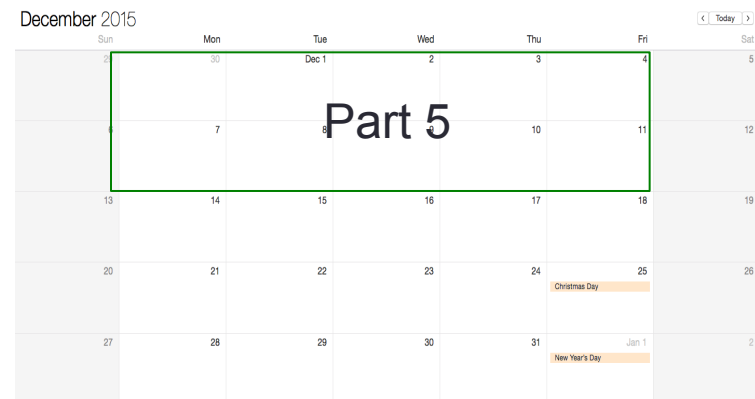
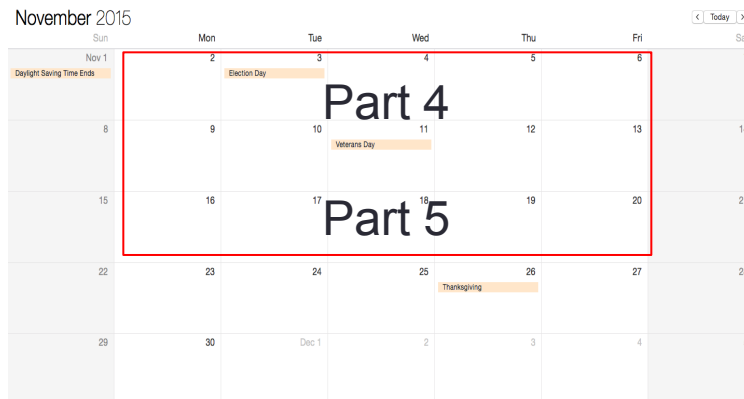
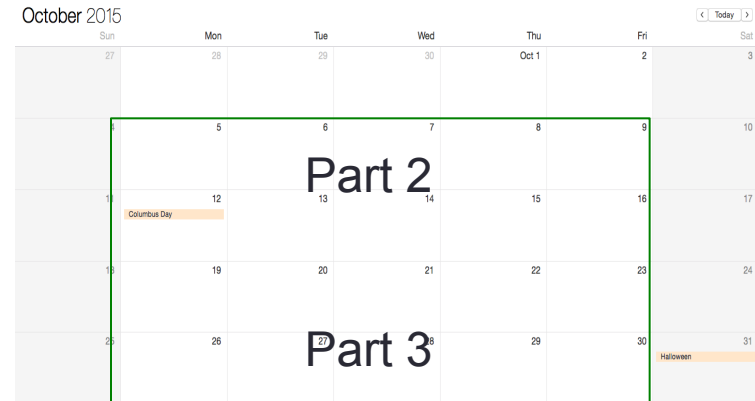
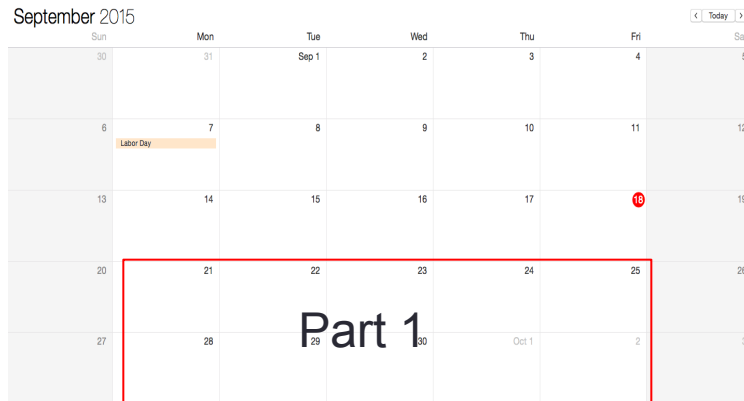
Syllabus

- Part 1 - Communicating with robot (2 weeks)
 - BLE communication and robot API
- Part 2 - Event Driven Behavior (2 weeks)
 - Finite State Machine (Behavior Tree)
- Part 3 - Reasoning with Uncertainty (2 weeks)
 - Dealing with noisy data, uncertainty in sensing and control
- Part 4 - Extending the robot (1 weeks)
 - I/O extensions: digital, analog, servo, pwm, etc
- Part 5 – Putting it together (including UI/UX) (3 weeks)
 - Design and implement of final (group) project
 - Encourage you to go “above and beyond”

Grading

- This class will be project base only
 - No exam 😊
- There will be 4 individual projects and 1 team (final) project
 - Project #1 (Communication) – 20%
 - 2 weeks
 - Project #2 (Finite State Machine) - 20%
 - 2 weeks
 - Project #3 (Uncertainty) - 20%
 - 2 weeks
 - Project #4 (Robot Extension) - 10%
 - 1 week
 - Project #5 (Final, Group project) - 30%
 - 3 weeks
 - Design your (team) project (need to get approval)

Calendar



KC
Teaching

David
Teaching

About US

- Instructors
 - Dr. Kyong-Sok “KC” Chang
 - Dr. David Zhu
- TA
 - Jocelyn Neff
 - Kornel Niedziela

Logistics

- Getting your own Hamster
 - Sign-up sheet
- Programming environment
 - Mac
 - PC
- Website for the class
- TA sessions (office hours)
 - Location
 - Time
- Emails

Outline

- BLE: Hamster test
- DRC: communication
- Robot communication protocol
- IoT communication protocol
- Bluetooth History / versions
- BLE Specifications
- BLE Protocol
- GAP: Generic Access Profile
- GATT: Generic Attribute Profile
- Assignment#1

Robot with computer or Computer with legs?



www.shutterstock.com - 25881154

BLE: Reading

This week's(and future) reading for BLE.

“Getting started with Bluetooth Low Energy” by Townsend, Davidson & Akiba,
O’Reilly

<https://www.safaribooksonline.com/library/view/getting-started-with/9781491900550/cover.html>

BLE: Hamster

Generic Apps: connecting to Hamster

- iPhone, iPad, Mac: LightBlue
- Android: nRF Master Control Panel

Sensors: (UUID: 0x00009001...)

-- Read data (20 bytes) from Hamster.

(in hex)

- 1st byte: version/topology
- 2nd byte: network ID
- 3rd byte: command/security
- 4th byte: Signal Strength (-128~0)
- 5th byte: Left Proximity (0~255)
- 6th byte: Right Proximity (0~255)
- 7th byte: Left Floor (0~255)
- 8th byte: Right Floor (0~255)

Effectors: (UUID: 0x0000A000...)

-- Write 11 bytes to Hamster.

<0000103232020300000040>

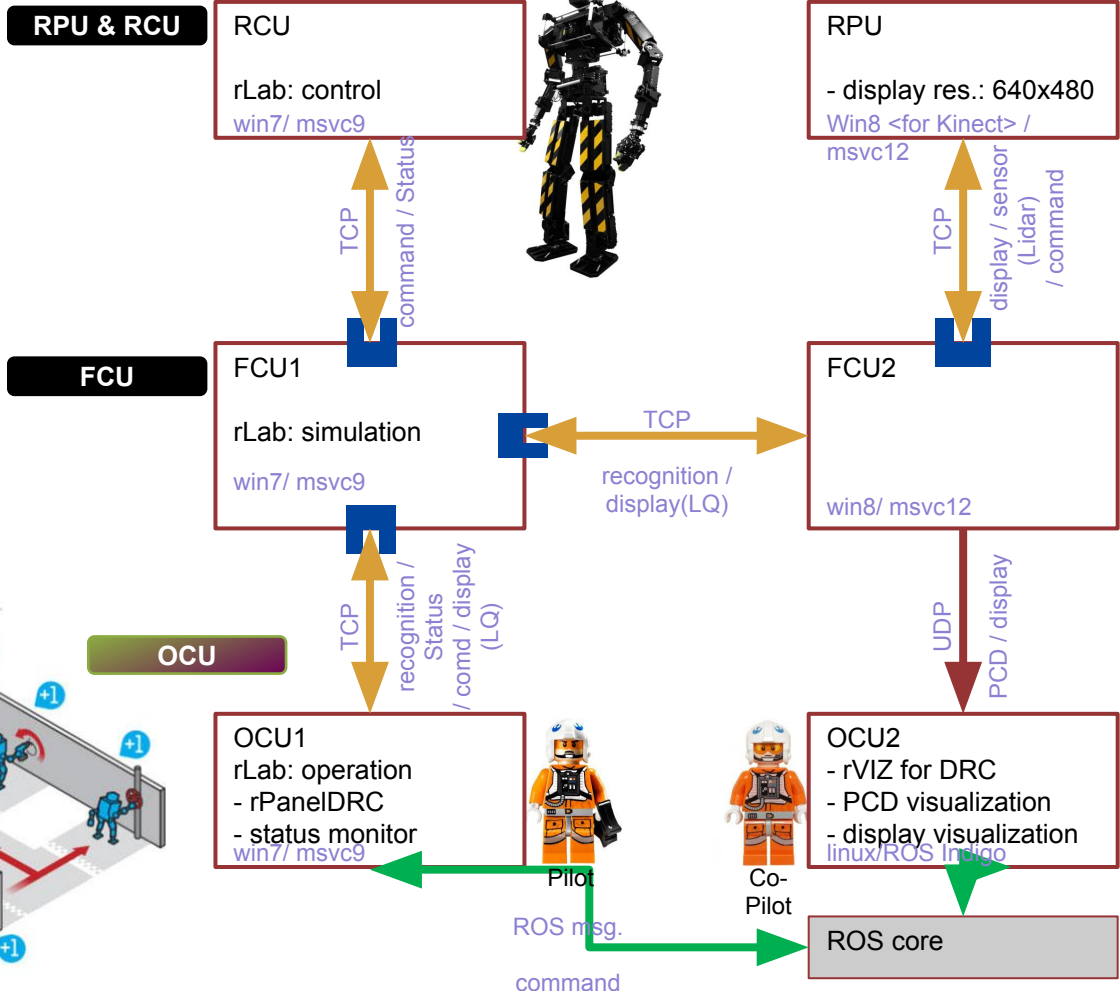
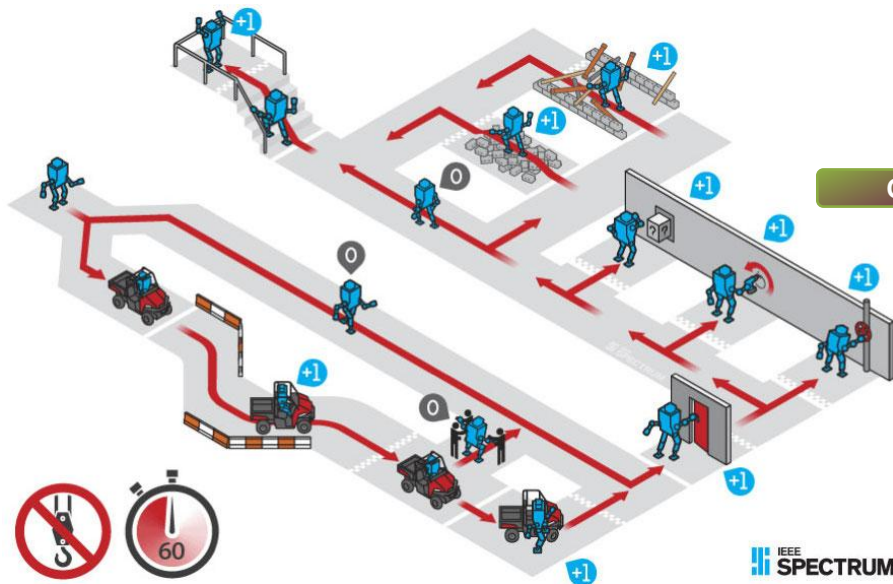
(in hex)

- 0x00: version/topology
- 0x00: network ID
- 0x10: command/security
- 0x32: left wheel speed (50: -100~100)
- 0x32: right wheel speed (50: -100~100)
- 0x02: left LED color (green: 0~7)
- 0x03: right LED color (blue: 0~7)
- 0x00: buzzer high
- 0x00: buzzer middle
- 0x00: buzzer low
- 0x40: musical note (C4: middle C: 0-88)

DRC: communication

Robot operation

- Autonomous (AI)
- Semi-autonomous
- Tele-operation



Robot Communication Protocol

- RS232
 - cheapest (many PCs have serial port)
 - 115.2 kbps
- CANopen (CAN)
 - many microcontrollers have built in CAN ports
 - 1 Mbps
 - error correction (collision detection/prevention)
 - daisy-chain (128 nodes)
- RS485
 - 32 nodes
 - 921.6 kbps
- EtherCAT
 - high-performance
 - 200 Mbps
 - based on Ethernet (no special Ethernet hardware)
 - Multiple topologies possible: line, star, tree, daisy-chain
 - Real-time down to the I/O level

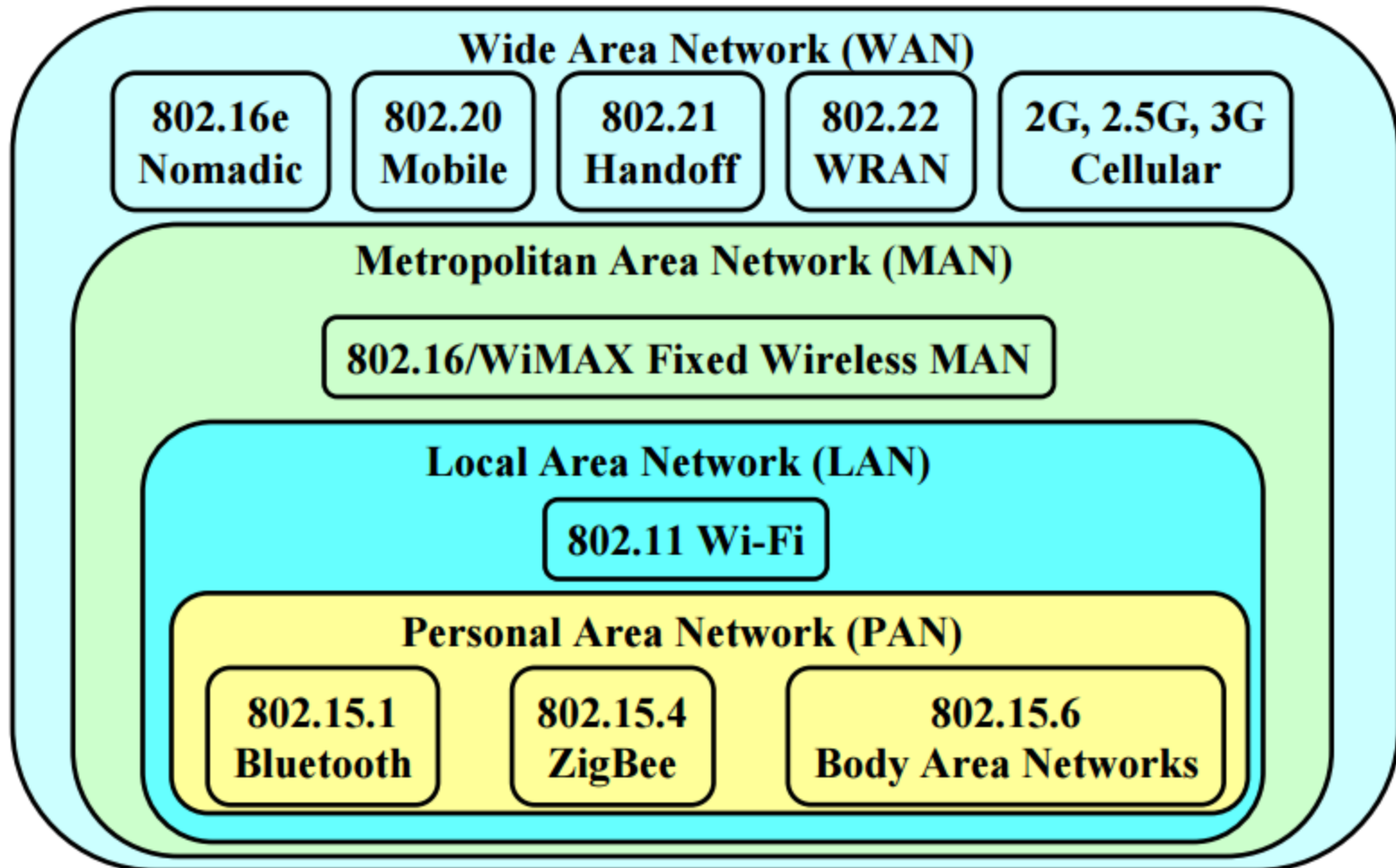
Robots: Bluetooth Smart Devices (That Can Move)



Bluetooth Smart Devices



IoT Communication Protocol



Ref.: Raj Jain, Washington University in St. Louis

Bluetooth: History


- Bluetooth Low Energy (BLE, also marketed as Bluetooth Smart) started as part of the Bluetooth 4.0 Core Specification.
- Started with Ericsson's (Sweden) Bluetooth Project in 1994 for radio communication between cell phones over short distances
- Named after Danish king Harald Blaatand (AD 940-981) who was fond of blueberries and had a blue tooth
- Intel, IBM, Nokia, Toshiba, and Ericsson formed Bluetooth SIG (special interest group) in May 1998. (Today it has membership of over 11,000 companies)
- Version 1.0A of the specification came out in late 1999.
- IEEE 802.15.1 approved in early 2002 is based on Bluetooth Later versions handled by Bluetooth SIG directly
- Key Features:
 - Lower Power: 10 μ A in standby, 50 mA while transmitting
 - Cheap: \$5 per device
 - Small: 9 mm² single chips

Bluetooth by Intel engineer Jim Kardach

Who is Bluetooth?

Harald Blaatand “Bluetooth” II

- ◆ King of Denmark 940-981
 - Son of Gorm the Old (King of Denmark) and Thyra Danebod (daughter of King Ethelred of England)
- ◆ This is one of two Runic stones erected in his capital city of Jelling (central Jutland)
 - This is the front of the stone depicting the chivalry of Harald
 - Harald controlled Denmark and Norway
 - Harald thinks mobile PCs and cellular phones should seamlessly communicate



Bluetooth October 1998 Bluetooth SIG Confidential

Bluetooth versions

- Bluetooth 1.0 and 1.0B (late 1999): too many problems
- Bluetooth 1.1: IEEE 802.15.1-2002. RSSI (received signal strength indicator, WPAN (wireless personal area network)
- Bluetooth 1.2: IEEE 802.15.1-2005 (Nov 2003): 721 kbs
- Bluetooth 2.0 + Enhanced Data Rate (EDR) (Nov 2004): 3 Mbps
- Bluetooth 2.1 + EDR (July 2007): Secure Simple Pairing to speed up pairing
- Bluetooth 3.0+ High Speed (HS) (April 2009): 24 Mbps using WiFi PHY + Bluetooth PHY for lower rates
- Bluetooth 4.0 (June 2010): Low energy. 1 Mbps. Smaller devices requiring longer battery life (several years). New incompatible PHY. Bluetooth Smart Ready, Bluetooth Smart or BLE
- Bluetooth 4.1 (Dec 2013): 4.0 + Core Specification Amendments (CSA) 1, 2, 3, 4, current standard
- Bluetooth 4.2 (Dec 2014): 4.1 + flexible internet connectivity + power efficiency + security + faster

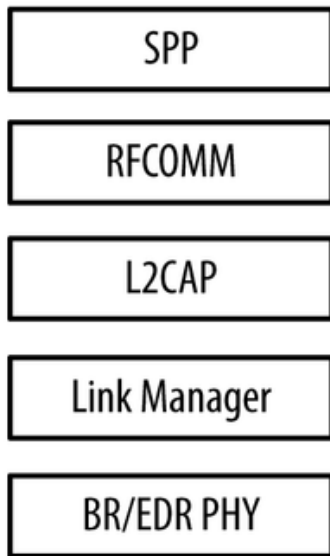
BLE: specification

- Range: ~ 150 meters open field
- Output Power: ~ 10mW (10dBm)
- Max Current: ~ 15mA
- Latency: 3 ms
- Topology: Star
- Connections: > 2 billion
- Modulation: GFSK @ 2.4 GHz
- Robustness: Adaptive Frequency Hopping, 24 bit CRC
- Security: 128bit AES CCM
- Sleep current ~ 1 μ A
- 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 optimised for file transfer.
 - It is designed for sending small chunks of data (exposing state).

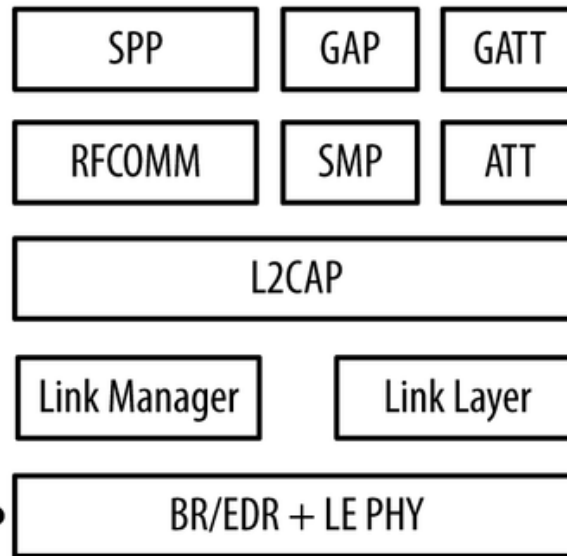
Bluetooth protocols



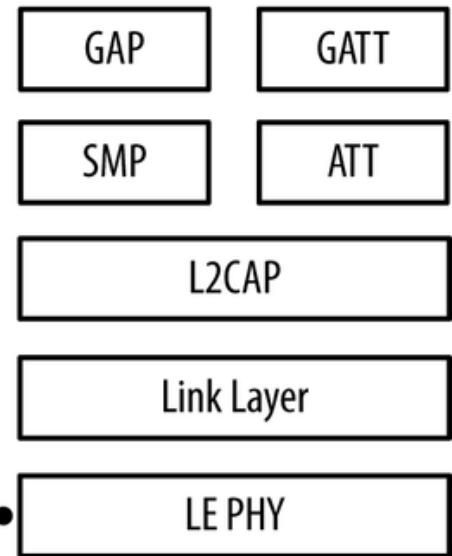
(classic or BR/EDR)



(dual mode or BR/EDR/LE)



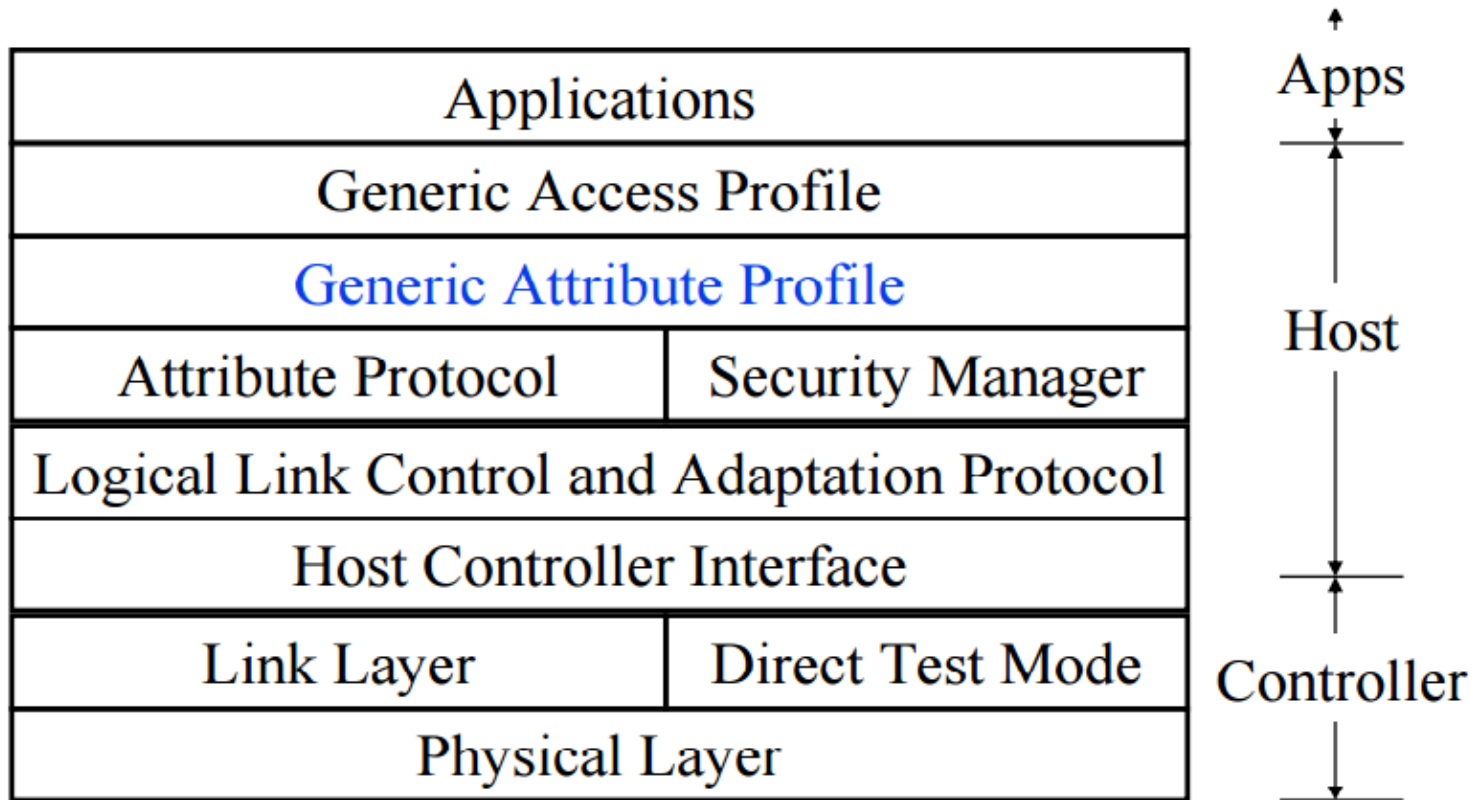
(single mode or BLE)



Configurations between Bluetooth versions and device types

Ref. "Getting started with Bluetooth Low Energy" by Townsend, Davidson & Akiba, O'Reilly

Bluetooth Smart Protocol Stack

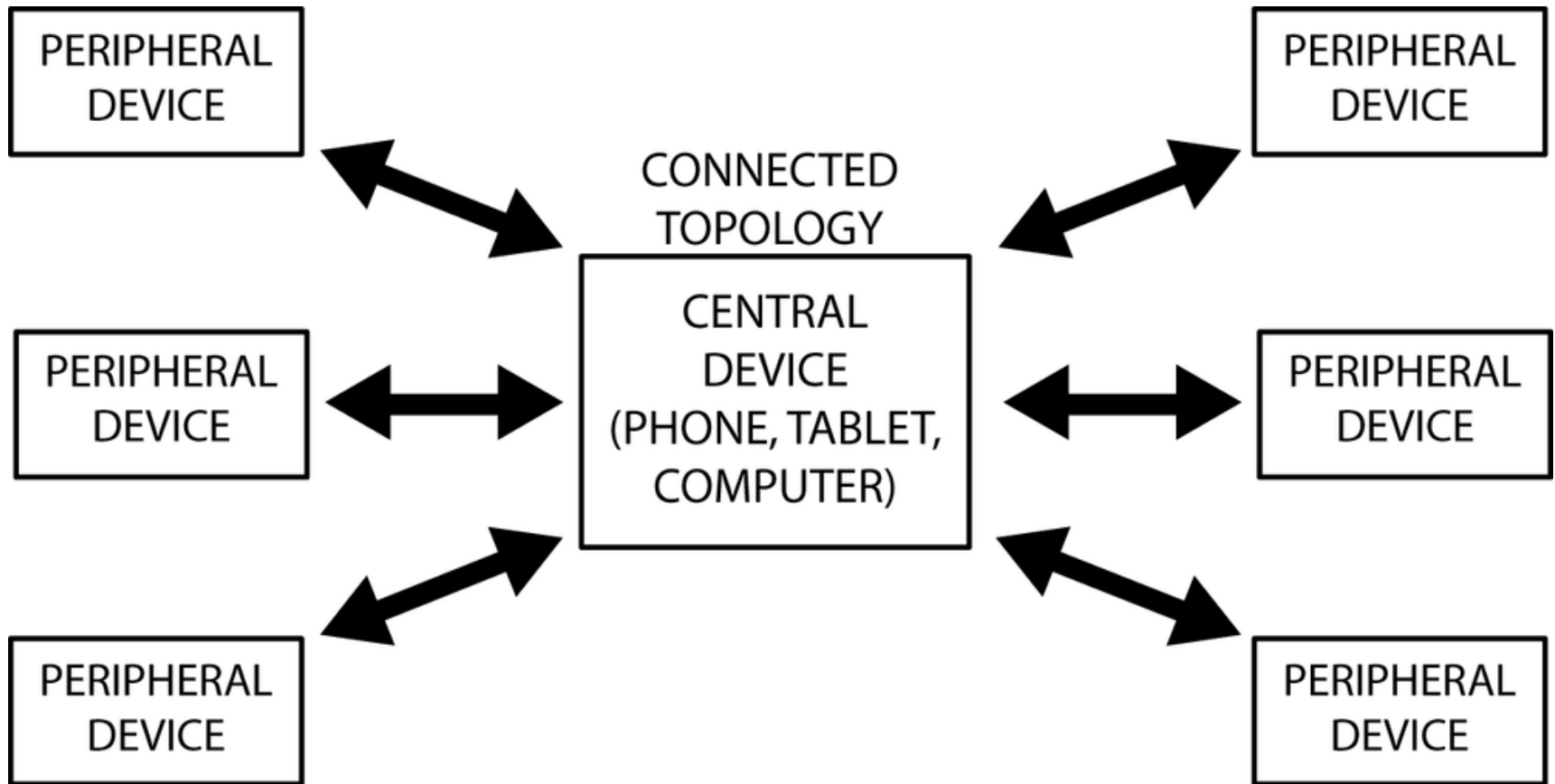


Ref. Raj Jain, Washington University in St. Louis

GAP

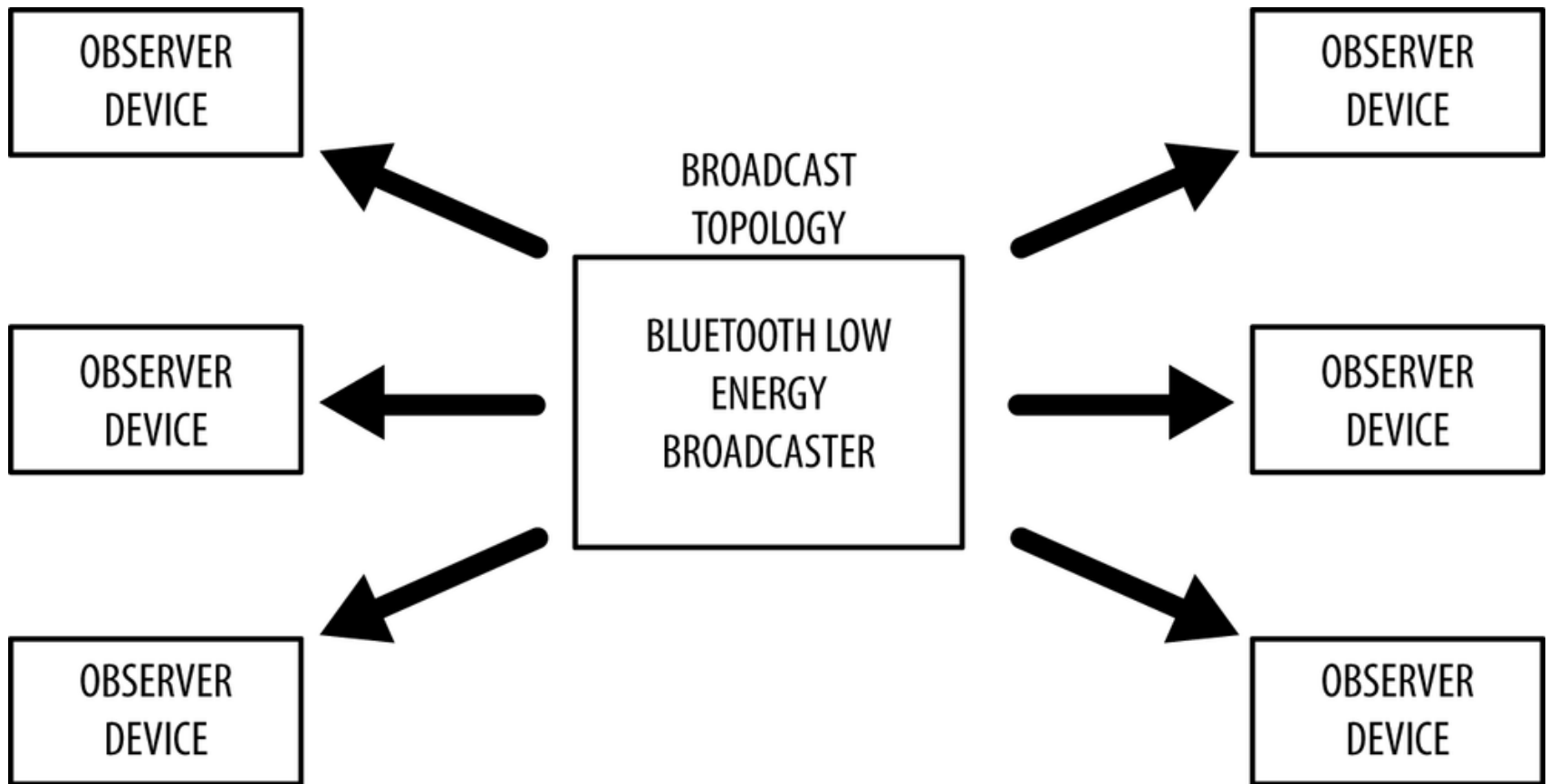
- Generic Access Profile (GAP)
 - defines lower-level interactions with devices (lower-level radio protocols)
 - broadcast, discover, establish connections, manage connections, negotiate security levels
 - only one master per slave
 - multiple slaves per master
 - in theory: no limit
 - in practice: up to 8 simultaneous connection
- Role: Master - Slave
- Master / Central
 - the BLE device which initiates an outgoing connection request to an advertising peripheral device
- Slave / Peripheral
 - the BLE device which accepts an incoming connection request after advertising
- Role: Observer - Broadcaster (Non-connecting)

Role: Central - Peripheral



Ref. "Getting started with Bluetooth Low Energy" by Townsend, Davidson & Akiba, O'Reilly

Role: Observer - Broadcaster



Ref. "Getting started with Bluetooth Low Energy" by Townsend, Davidson & Akiba, O'Reilly

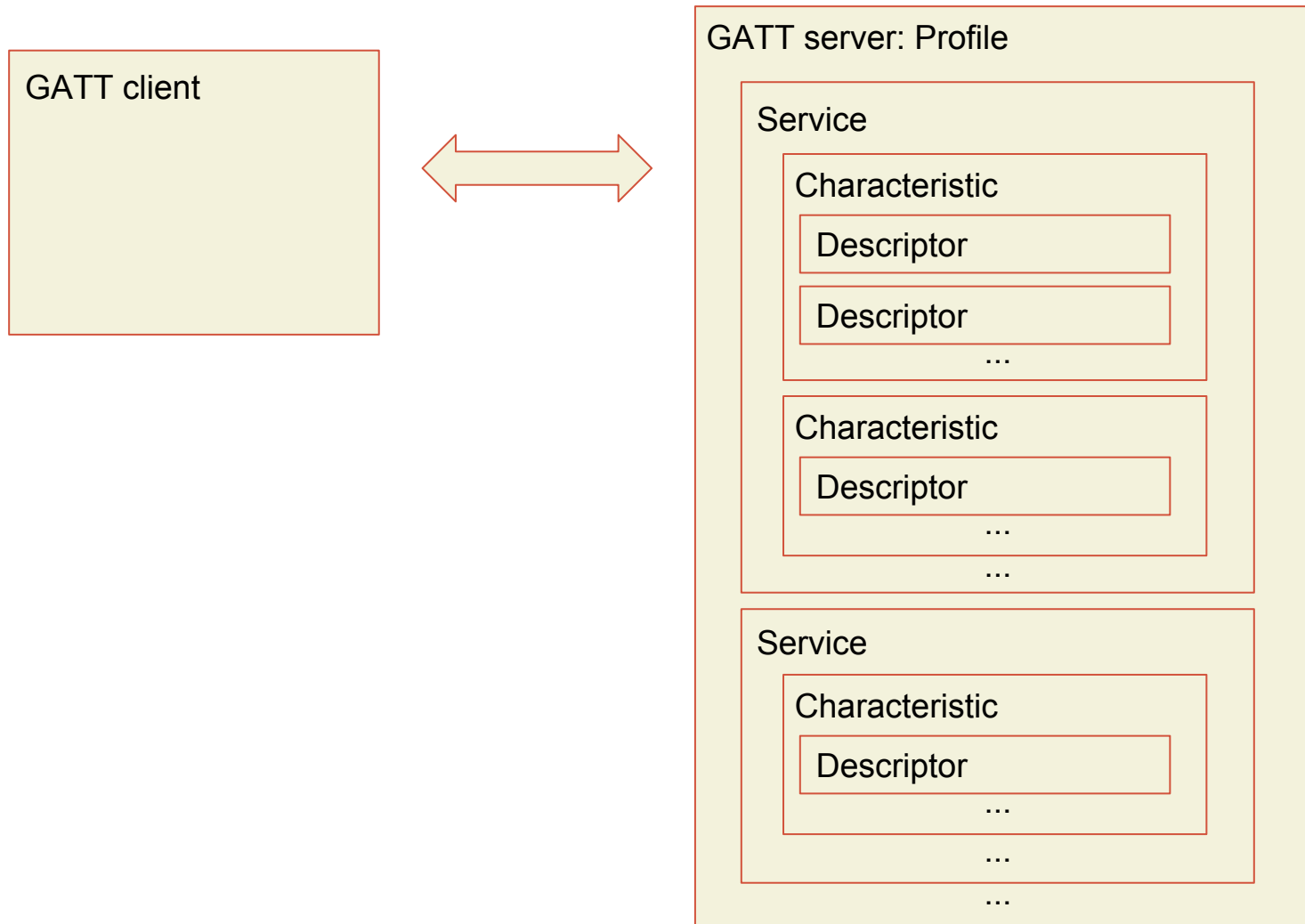
GATT

- Generic Attribute Profile (GATT)
 - how to exchange all profiles and user data over BLE
 - deals with actual data transfer procedures and formats: discover, read, write, and push data elements
 - provides reference framework for GATT-based profiles
 - SIG defined profiles
 - Custom profiles
 - uses Attribute Protocol (ATT) to exchange data between devices
- Role: Client - Server (independent of Master-Slave role)
 - GATT client
 - a device which accesses data on the remote GATT server via read, write, notify, or indicate operations
 - GATT server
 - a device which stores data locally and provides data access methods to a remote GATT client

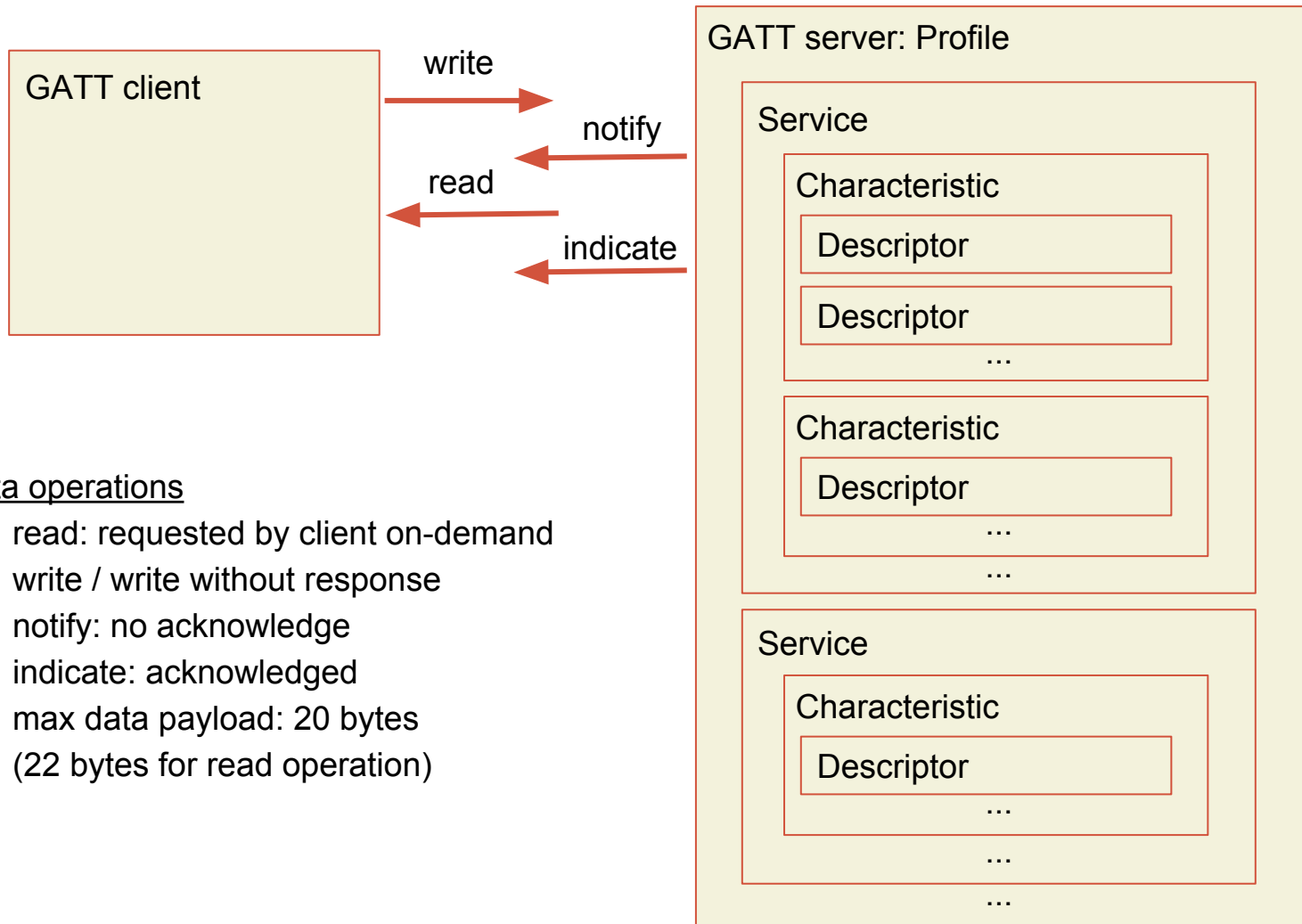
GATT

- UUIDs
 - A universally unique identifier (UUID) is a 128-bit (16 bytes) number that is guaranteed (or has a high probability) to be globally unique.
 - two additional UUID formats: 16-bit and 32-bit UUIDs.
 - Bluetooth SIG as standard Bluetooth UUIDs:
 - To reconstruct the full 128-bit UUID from the shortened version
 - xxxxxxxx-0000-1000-8000-00805F9B34FB
 - Custom UUID (vendor-specific UUID): 128-bit UUID: generated using ITU's UUID generation page.
- Attributes
 - Every characteristic has one main attribute which allows access to the actual value stored in the database for that characteristic.
 - A characteristic can have multiple attributes.
 - Each attribute has a unique UUID.
- GATT server
 - must implement the official **Generic Access service** (0x1800)
 - Two mandatory characteristics: **Device Name** (0x2A00) and **Appearance** (0x2A04)
 - Similar to **Friendly Name** and **Class of Device** values used by classic Bluetooth.

GATT: Data Hierarchy



GATT: Data Transfer Methods



4 data operations

- read: requested by client on-demand
- write / write without response
- notify: no acknowledge
- indicate: acknowledged
- max data payload: 20 bytes
(22 bytes for read operation)

GAP Profile: Hamster

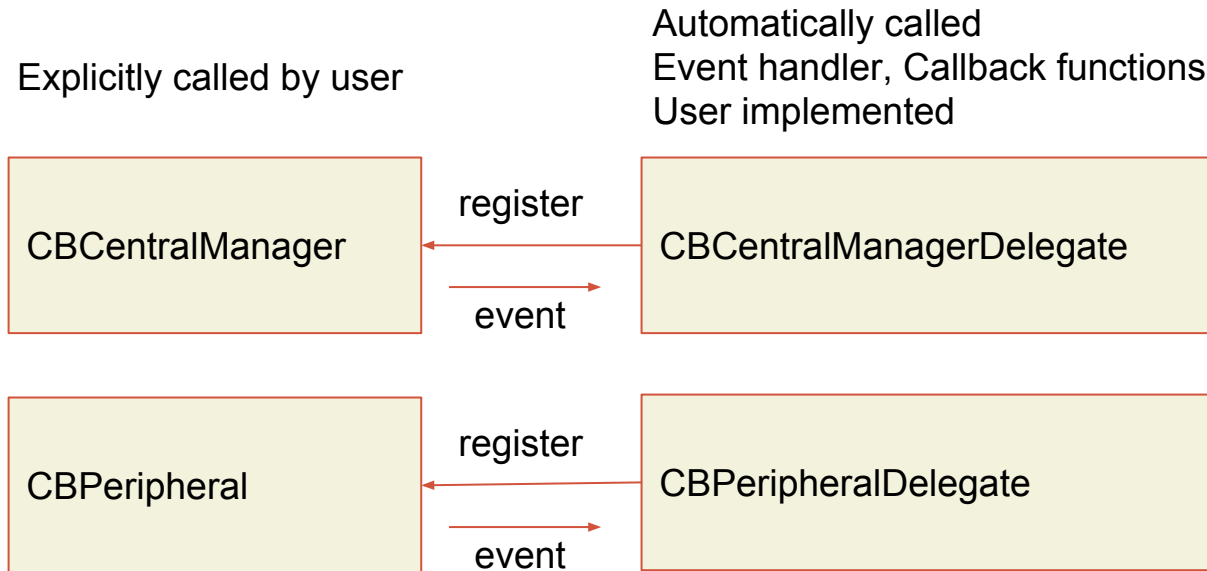
- [-] Hamster (0xFC11FF7E6ADD) (-62dBm)
 - ... RSSI: -62dBm
 - ... Address: FC11FF7E6ADD
 - ... Address Type: Random
 - ... Advertising Type: Connectable
 - ... Bonded: False
 - [-] Advertising Data
 - ... CompleteLocalName: Hamster
 - ... Appearance: 0x0402
 - ... Flags: GeneralDiscoverable, BrEdrNotSupported
 - ... ServicesCompleteListUuid16: 0xF138
 - ... Scan Response Data

GATT Profile: Hamster

- PrimaryService, Value: 00-18, Generic Access (0x1800)
 - CharacteristicDeclaration, Value: 0A-03-00-00-2A, Properties: Read, Write, Characteristic UUID: 0x2A00
 - DeviceName, Value: 48-61-6D-73-74-65-72, DeviceName: Hamster
 - CharacteristicDeclaration, Value: 02-05-00-01-2A, Properties: Read, Characteristic UUID: 0x2A01
 - Appearance, Value: 02-04, Appearance: 0x0402
 - CharacteristicDeclaration, Value: 02-07-00-04-2A, Properties: Read, Characteristic UUID: 0x2A04
 - SlavePreferredConnectionParameters, Value: 10-00-10-00-00-00-32-00, MinConnInterval: 0x0010, MaxConnInterval: 0x0010, SlaveLatency: 0x0000, SupervisionTimeoutMultiplier: 0x0032
- PrimaryService, Value: 01-18, Generic Attribute (0x1801)
- PrimaryService, Value: 0A-18, Device Information (0x180A)
 - CharacteristicDeclaration, Value: 02-0B-00-29-2A, Properties: Read, Characteristic UUID: 0x2A29
 - Manufacturer Name String, Value: 52-6F-62-6F-6D-61-74-69-6F-6E
 - CharacteristicDeclaration, Value: 02-0D-00-26-2A, Properties: Read, Characteristic UUID: 0x2A26
 - Firmware Revision String, Value: 31-2E-32
- PrimaryService, Value: 66-9A-0C-20-00-08-E2-A5-E3-11-80-9C-01-90-00-00, 0x00009001-9C80-11E3-A5E2-0800200C9A66
 - CharacteristicDeclaration, Value: 14-10-00-66-9A-0C-20-00-08-E2-A5-E3-11-80-9C-0A-90-00-00, Properties: WriteWithoutResponse, Notify, Characteristic UUID: 0x0000900A-9C80-11E3-A5E2-0800200C9A66
 - UUID: 0000900A-9C80-11E3-A5E2-0800200C9A66, Value: 00-00-10-00-1B-00-65-14-02-BF-12-BF-C7-7F-00-00-3C-F9-F8-00
 - CharacteristicUserDescription, Value: 53-65-6E-73-6F-72-73, UserDescription: Sensors
 - ClientCharacteristicConfiguration, Value: 00-00, CharacteristicConfigurationBits: None (0x0000)
- PrimaryService, Value: 66-9A-0C-20-00-08-E2-A5-E3-11-80-9C-00-A0-00-00, 0x0000A000-9C80-11E3-A5E2-0800200C9A66
 - CharacteristicDeclaration, Value: 04-15-00-66-9A-0C-20-00-08-E2-A5-E3-11-80-9C-06-A0-00-00, Properties: WriteWithoutResponse, Characteristic UUID: 0x0000A006-9C80-11E3-A5E2-0800200C9A66
 - UUID: 0000A006-9C80-11E3-A5E2-0800200C9A66, Value: 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
 - CharacteristicUserDescription, Value: 45-66-66-65-63-74-6F-72-73, UserDescription: Effectors
- PrimaryService, Value: 23-D1-BC-EA-5F-78-23-15-DE-EF-12-12-30-15-00-00, DFU (0x00001530-1212-EFDE-1523-785FEABCD123)
 - CharacteristicDeclaration, Value: 04-19-00-23-D1-BC-EA-5F-78-23-15-DE-EF-12-12-32-15-00-00, Properties: WriteWithoutResponse, Characteristic UUID: 0x00001532-1212-EFDE-1523-785FEABCD123
 - DFU Packet, (No values read)
 - CharacteristicDeclaration, Value: 18-1B-00-23-D1-BC-EA-5F-78-23-15-DE-EF-12-12-31-15-00-00, Properties: Write, Notify, Characteristic UUID: 0x00001531-1212-EFDE-1523-785FEABCD123
 - DFU Control Point, (No values read)
 - ClientCharacteristicConfiguration, Value: 00-00, CharacteristicConfigurationBits: None (0x0000)
 - CharacteristicDeclaration, Value: 02-1E-00-23-D1-BC-EA-5F-78-23-15-DE-EF-12-12-34-15-00-00, Properties: Read, Characteristic UUID: 0x00001534-1212-EFDE-1523-785FEABCD123
 - UUID: 00001534-1212-EFDE-1523-785FEABCD123, Value: 01-00

Assignment#1-1

Event-driven programming



Ex) By calling

`CBCentralManager::scanForPeripheralsWithServices_options_([Ads], None),`

`CBCentralMangerDelegate::`

`centralManager_didDiscoverPeripheral_advertisementData_RSSI_(self,`

`manager, peripheral, data, rssi)`

will be called automatically.

Assignment#1-1: Due 10/01/2015

Hints:

In OS X, CentralManager == host/client/computer, Peripheral == slave/server/device(Hamster).

File: "hamsterAPI_ref.py"

Look at the required methods for delegates and available class methods in "hamsterAPI_ref.py".

Notice that this file is not a correct python file. This file lists relevant APIs that you should use.

--You need to implement the methods for both CBCentralManagerDelegate and CBPeripheralDelegate protocols.

--And you need to use CBCentralManager class methods and CBPeripheral class methods in delegate methods.

***** Note: this is event driven programming such that once you call the CentralManager class methods or Peripheral class methods, corresponding delegate methods are being called automatically.

Ex) By calling CBCentralManager::scanForPeripheralsWithServices_options_([Ads], None),

CBCentralMangerDelegate::centralManager_didDiscoverPeripheral_advertisementData_RSSI_(self, manager, peripheral, data, rssi) will be called automatically.

Reference:

Mac OS X's Core Bluetooth Framework Reference

https://developer.apple.com/library/prerelease/ios/documentation/CoreBluetooth/Reference/CoreBluetooth_Framework/index.html#//apple_ref/doc/uid/TP40011295

https://developer.apple.com/library/prerelease/ios/documentation/CoreBluetooth/Reference/CoreBluetooth_Framework/index.html#//apple_ref/doc/uid/TP40011295

https://developer.apple.com/library/prerelease/ios/documentation/CoreBluetooth/Reference/CoreBluetooth_Framework/index.html#//apple_ref/doc/uid/TP40011295

PyObjC that is a bridge between Python and Objective-C.

<https://pythonhosted.org/pyobjc/core/intro.html>

Reference

“Getting started with Bluetooth Low Energy” by Townsend, Davidson & Akiba, O’Reilly

<https://www.safaribooksonline.com/library/view/getting-started-with/9781491900550/cover.html>

BLE master/slave, GATT client/server, and data RX/TX basics by Jeff Rowberg

<https://bluegiga.zendesk.com/entries/25053373--REFERENCE-BLE-master-slave-GATT-client-server-and-data-RX-TX-basics>

<https://developer.bluetooth.org/TechnologyOverview/Pages/Technology-Overview.aspx>

http://www.cse.wustl.edu/~jain/cse574-14/ftp/j_11ble.pdf by Raj Jain

<http://chapters.comsoc.org/vancouver/BTLER3.pdf>