Bluetooth LE 4.0 and 4.1 (BLE)

Lab 11 Lunch
April 23rd, 2014

Noah Klugman
Josh Adkins

Outline

- History of Bluetooth
- Introduction to BLE
- Architecture
  - Controller
  - Host
  - Applications
- Power
- Topology
- Example: Heartbeat Sensor/App
- Competing formats
- Citations

History of Bluetooth

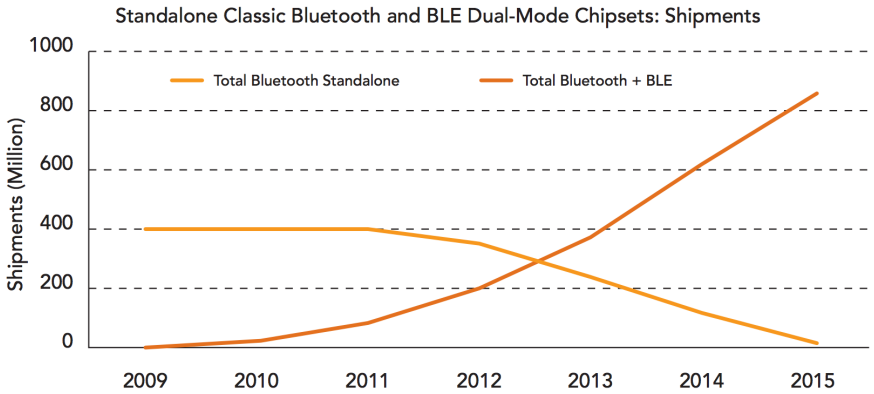
- First specification developed in 1994 by Ericsson as a cable replacement
  - 2.4 GHz ISM
  - Named after King Harold Bluetooth of Denmark who helped unify warring factions
- Bluetooth Special Interest Group (SIG) formed in 1998
  - No licence fee (although companies can still charge you…)
- 7 specifications released since 1998
  - All backward compatible (except for BLE)
- Billions and billions of devices

Outline

- History of Bluetooth
- Introduction to BLE
- Architecture
  - Controller
  - Host
  - Applications
- Power
- Topology
- Example: Heartbeat Sensor/App
- Competing formats
- Citations

Introduction to BLE

- New:
  - Radio
  - Protocol stack
  - Architecture
  - Qualification engine
- **But**: not backward compatible with Bluetooth Classic (including Bluetooth 4.0 Classic)



world [6]

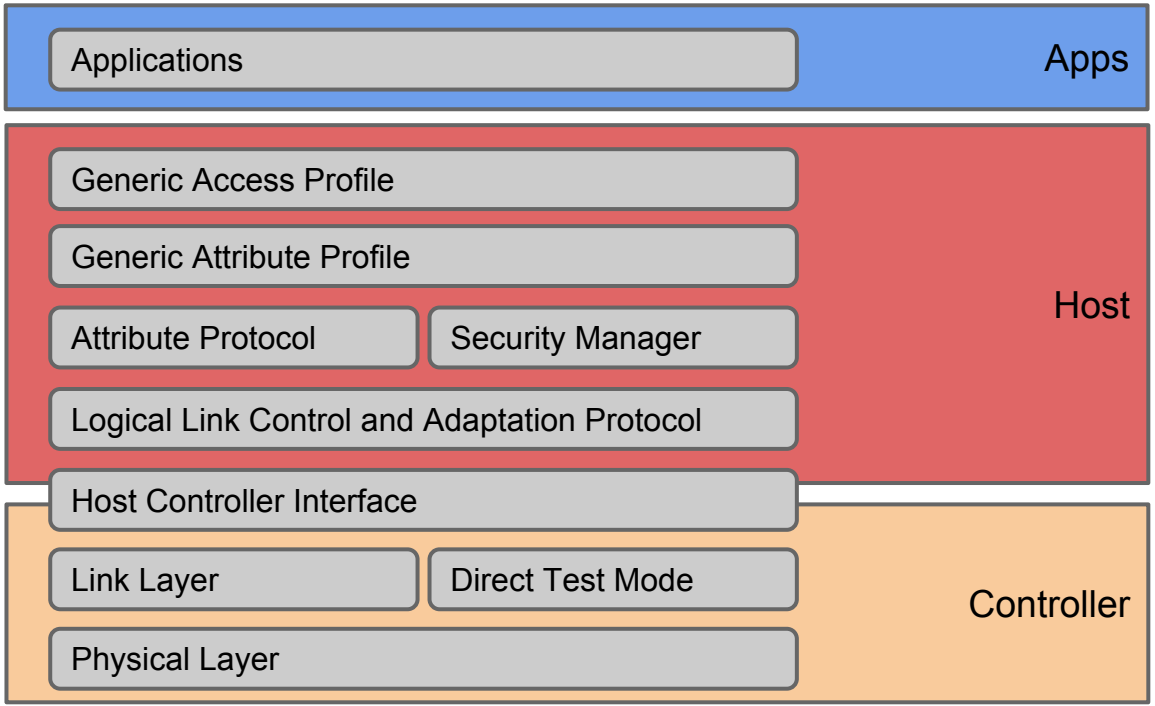
Introduction to BLE

- Low latency connection (3ms)
- Low power (15ma peak transmit, 1uA sleep)
  - Designed for coin cells
- Designed to send small packets of data (opposed to streaming)
  - Connect->transmit->disconnect->sleep
- Security
  - 128bit AES CCM
- Modulation
  - GFSK @ 2.4 GHz
- Adaptive Frequency Hopping
- 24 bit CRC
- Output Power: ~ 10mW (10dBm)

Outline

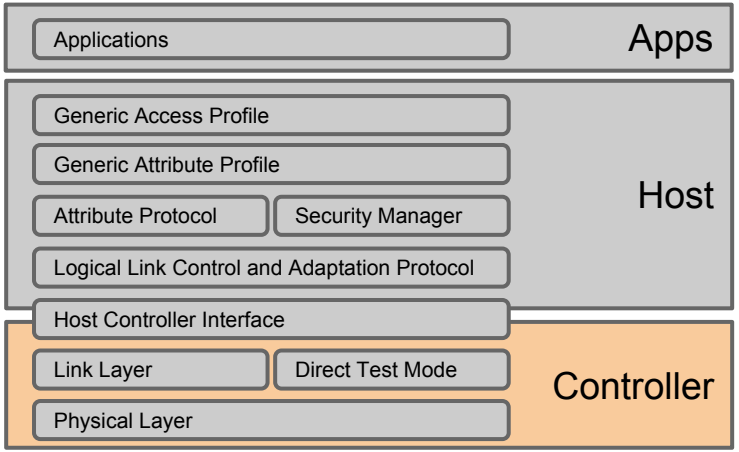
- History of Bluetooth
- Introduction to BLE
- Architecture
  - Controller
  - Host
  - Applications
- Power
- Topology
- Example: Heartbeat Sensor/App
- Competing formats
- Citations

Architecture



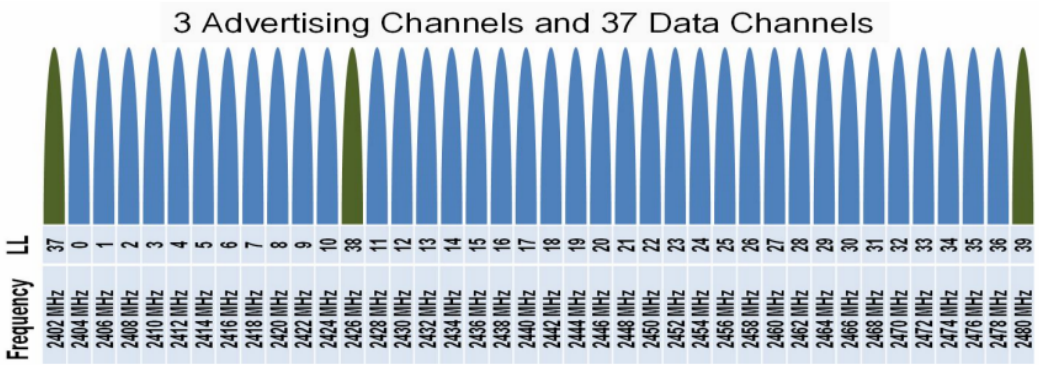
Architecture: Controller

- Radio Control
- Connection Logistics / Linking
- Radio Testing
- Interface to Host

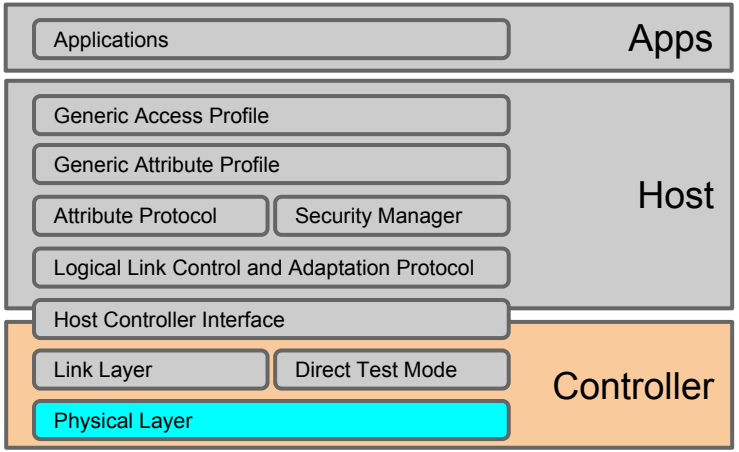


Architecture: Controller::Physical Layer

- 2.4 GHz ISM band
- 1 Mbps GFSK
- 40 Channels - 2MHz spacing
- Frequency Hopping in connections
  - Pseudo-random
  - Set in connection request
- Transmit power
  - -20 to +10dBm
- Receive sensitivity
  - -70 dBm



Core [5]



Architecture: Controller::Link Layer Terminology

- Transmit only -> advertiser
- Receive only -> scanner
- Bidirectional advertiser -> advertiser
- Bidirectional listener -> initiator
- Non-connected states use whitelist to prevent host wakeup

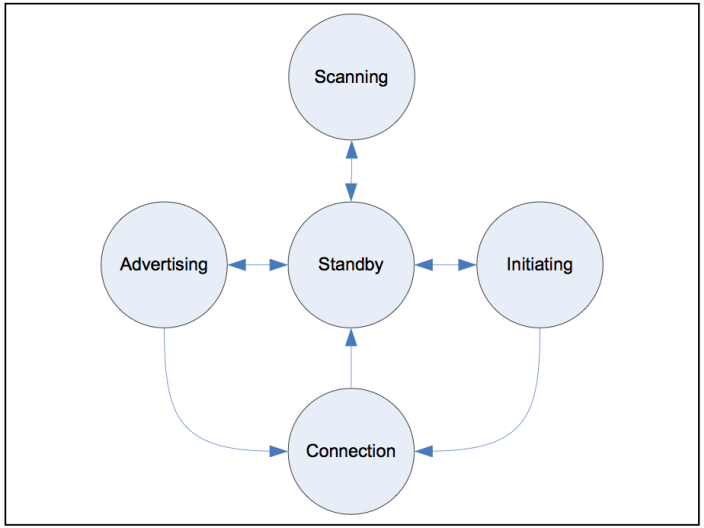
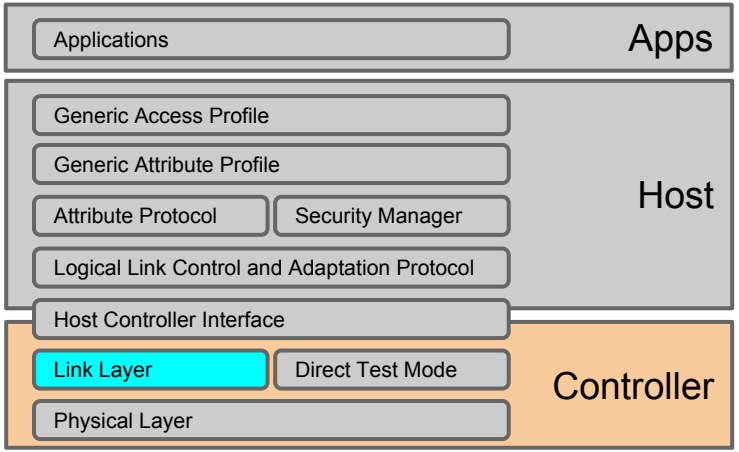


Figure 1.1: State diagram of the Link Layer state machine

Core [5]



Architecture: Controller::Link Layer Terminology (BLE 4.0)

- Master
  - can have multiple slaves
  - determines when slaves listen
  - determines frequency hopping algorithm
  - sends connection determination at connection request, but can update parameters after connection
  - if received packet from slave, need not respond
- Slave
  - only one master
  - if received packed from master, must respond

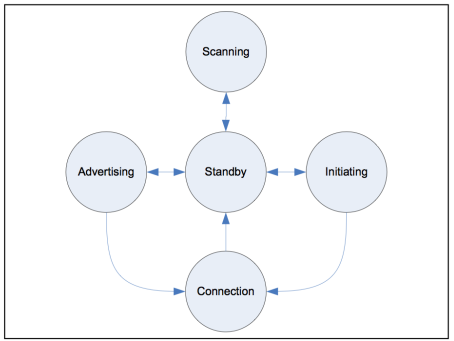
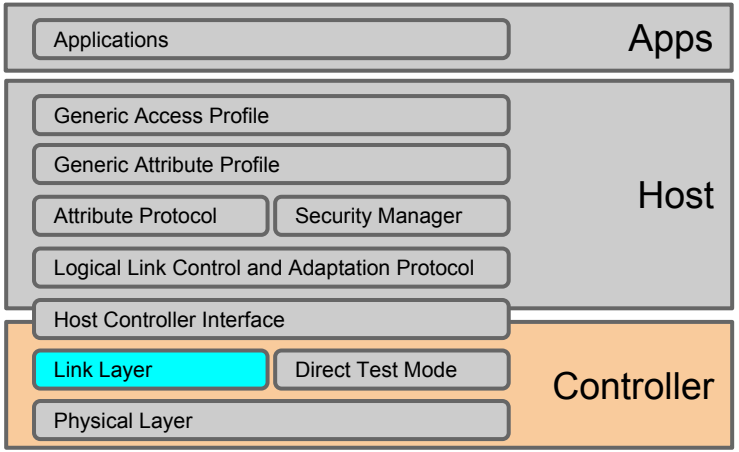


Figure 1.1: State diagram of the Link Layer state machine

Core [5]



Architecture: Controller::Link Layer (BLE 4.0)

- Upon Advertisement
  - When “advertisement event” interval hits all advertisements packets sent
- Upon Connection Attempted
  - Initiator transmits all connection parameters to advertiser in Connection Request
- Upon Connection
  - Physical layer divided into “connection events” at interval
  - In a connection event all packets are on same frequency
  - Master initiates all connection events
  - Connection can be closed or kept open normally, by request or at error

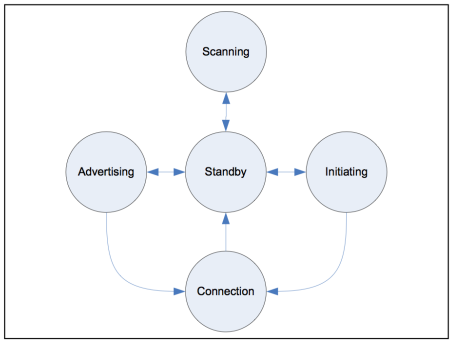
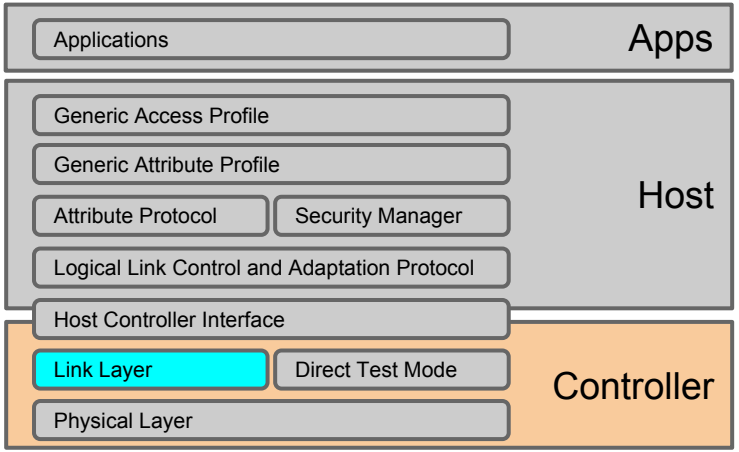
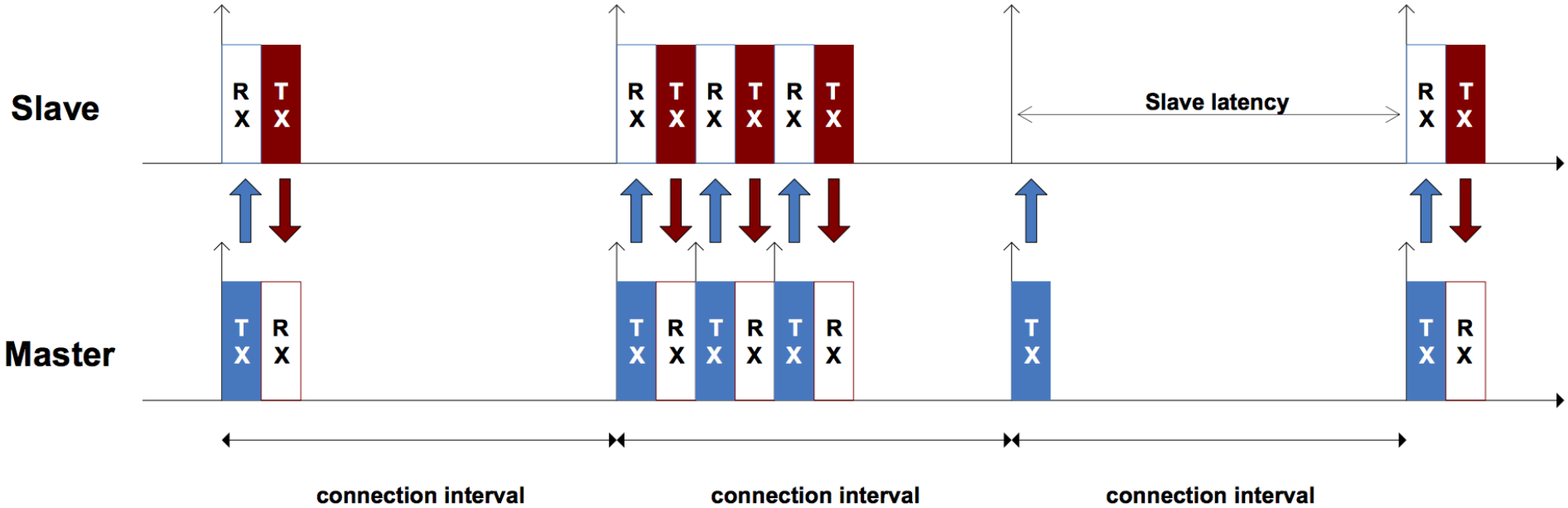


Figure 1.1: State diagram of the Link Layer state machine

Core [5]



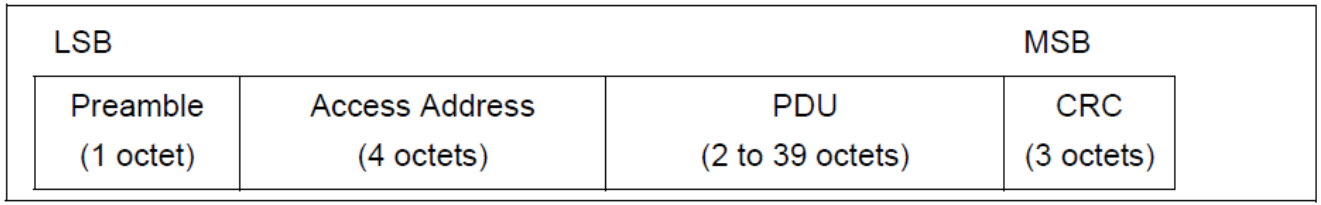
Architecture: Controller::Link Layer BLE 4.0



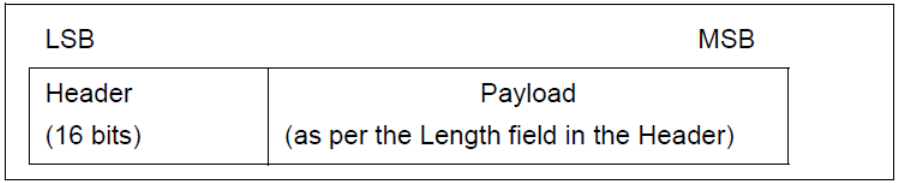
Nordic [3]

Architecture: (stepping back a bit…) Advertising Channel Packet Structure

Generic Packet Structure



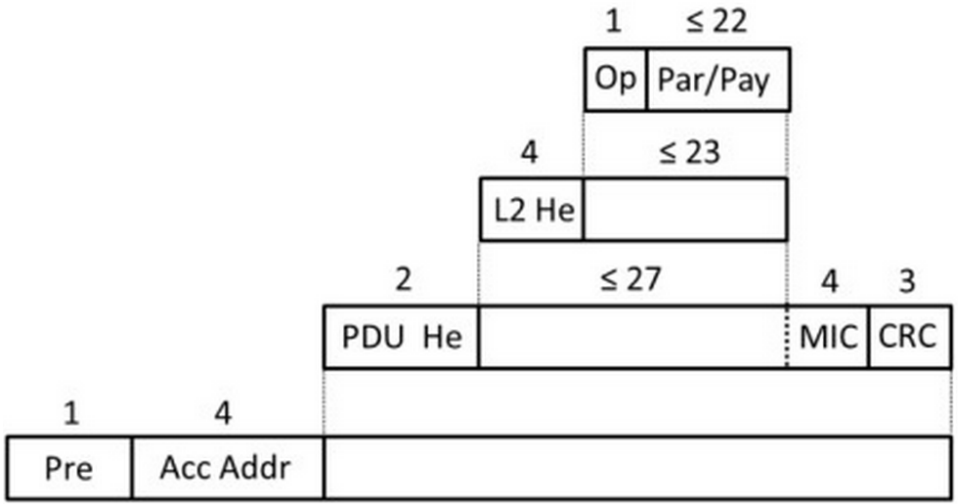
Generic Advertising PDU



Core [5]

- 7 Advertising Channel PDU Types
  - 4 Advertising, 2 Scanning, 1 Connect-request
  - Each has it's own payload specification
    - At least one address and possibly custom data

Architecture: (stepping back a bit…) Data Channel Packet Structure

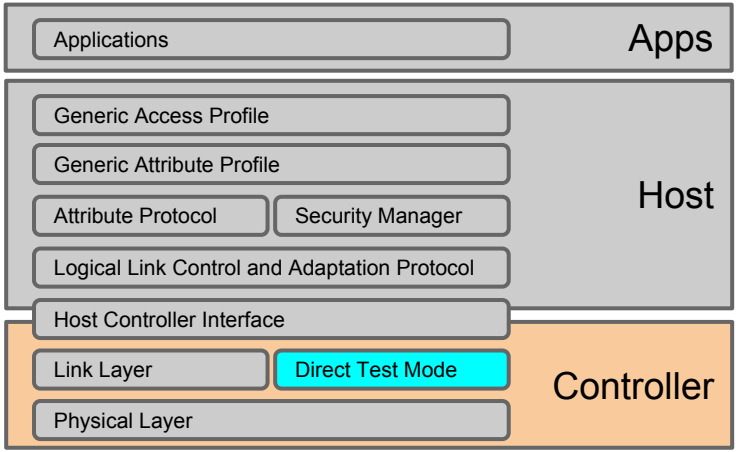


Par/Pay:	Parameters and Payload
Op:	ATT Opcode
PDU He:	PDU Header
L2 He:	L2CAP Header
Acc Addr:	Access Address
Pre:	Preamble
MIC:	Message Integrity Check
CRC:	Cyclic Redundancy Check

Core [5]

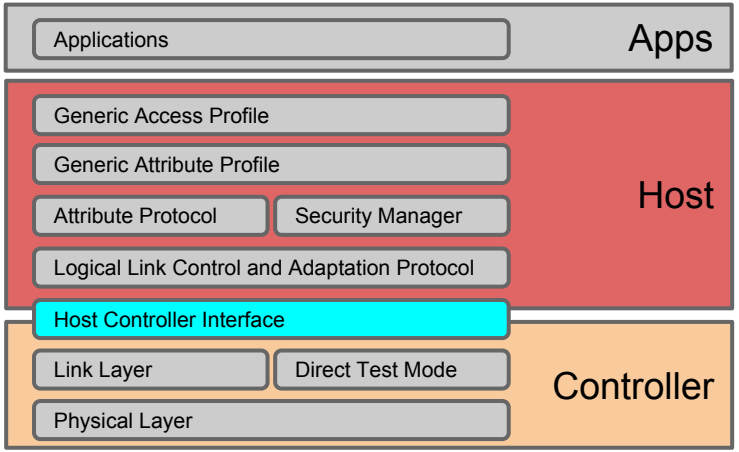
Architecture: Controller::Direct Test Mode

- Used for end-product qualification of RF transaction layer
  - All hardware must access directly using Host Controller Interface or 2-wire UART
- Transmit test mode:
  - test packets are generated
- Receive test mode:
  - counts number of test packets
- Standards of test packets and procedure available in spec



Architecture: Controller::Host Controller Interface (HCI)

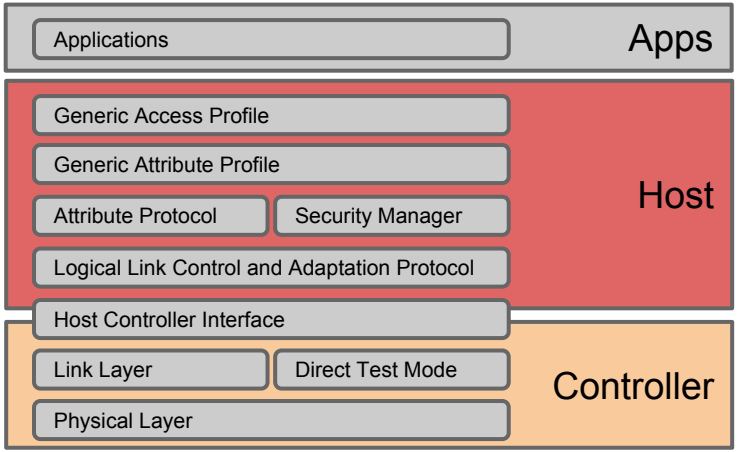
- Transport between between host and controller
  - allows changes between split layers
- Optional additional transport layer of UART, USB, 3-wire
- Accesses baseband, link manager commands and registers



HCI Interface (BLE CORE 4.1)

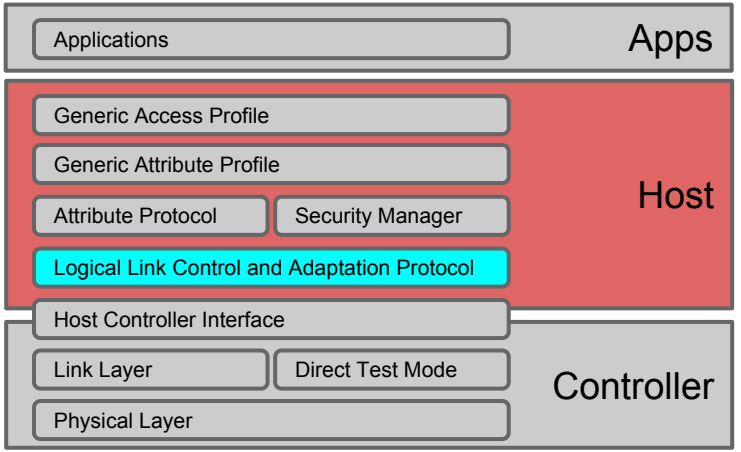
Architecture: Host

- Sits on top of the Radio
- Provides API to applications
- Much more relaxed timing



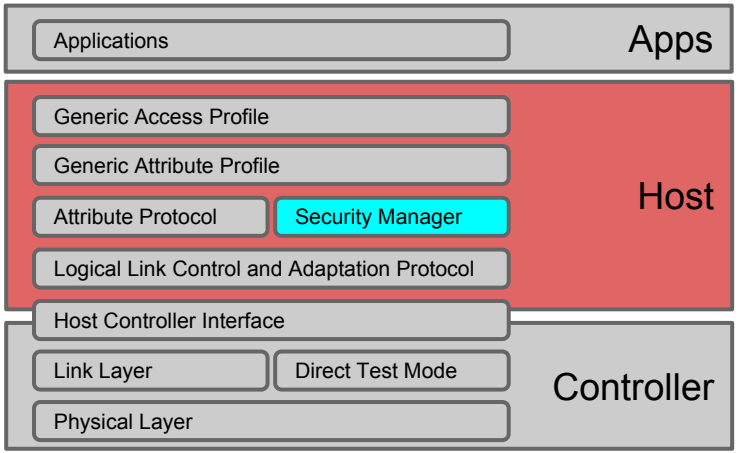
Architecture: Host::L2CAP

- Multiplexes between Attribute Protocol (ATT), Security Manager (SMP) and Link Layer controls through HCI
- Creates “channels” to logical layer
- Frame-oriented asynchronous and isosynchronous transport, negotiated by channel
- Error detection
- Pretty complicated part of the stack, punting a bit here...
  - Although it is simplified from earlier Bluetooth by not implementing flood control or retransmission to save power
- Backend for GAP



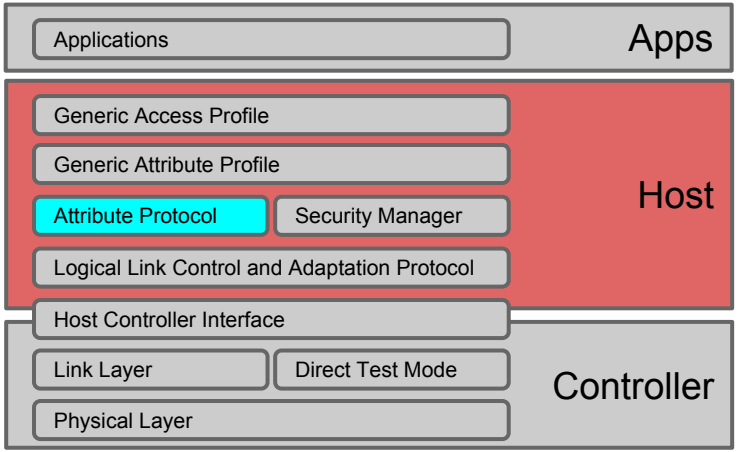
Architecture: Host::Security Manager

- Three phase process on connection
  - pairing feature exchange
  - short term key generation
  - transport specific key distribution (optional)
- Implements a number of cryptographic functions
- Memory and processing requirements are lower for responding
  - saves power



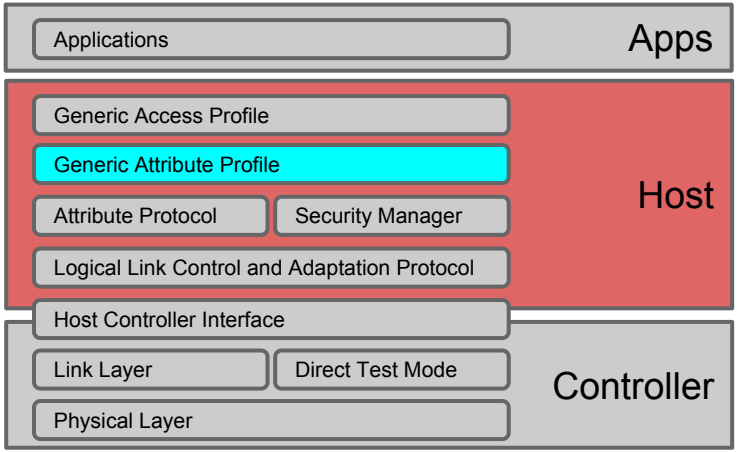
Architecture: Host::Attribute Protocol (ATT)

- Client Server Architecture
  - ATT server stores and serves data, client requests
  - Exposes data as a “attribute”
- Attribute
  - 16-bit handle used by client to address attribute
  - UUID: defines type and set by GAP and GATT
  - Value: length up *to* 512 octets
  - *permissions*: r/w/auth

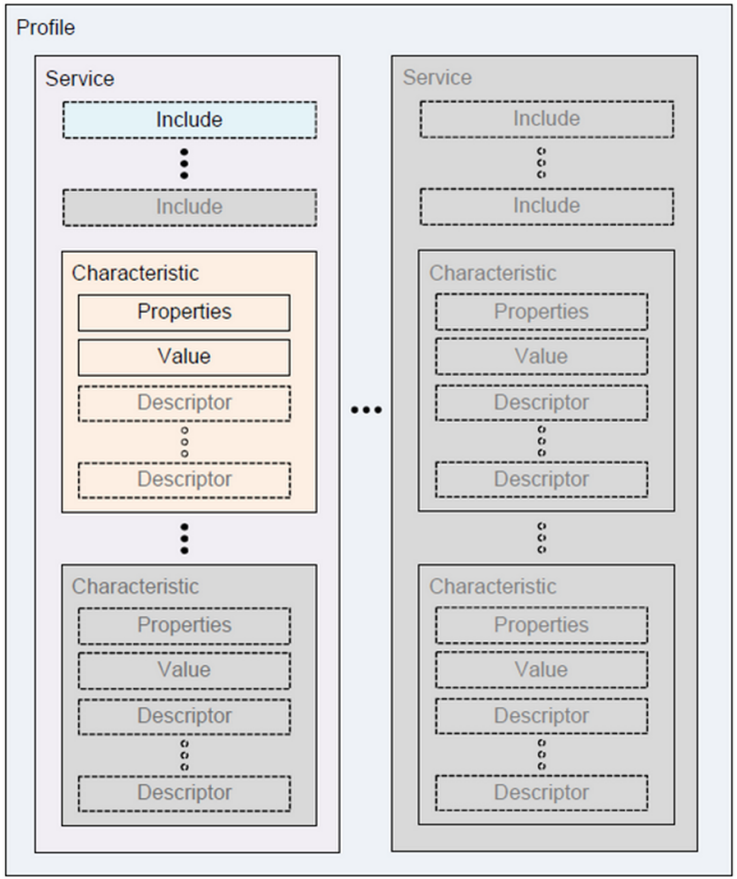


Architecture: Host::Generic Attribute Profile (GATT)

- Client Server Architecture (built on top of ATT)
  - Gatt Server stores data using ATT
  - Gatt Server accepts ATT requests to serve and save attributes
- Characteristics
  - Set of related attributes
  - One value, *n* descriptors
  - Exposes: features available, handle, representation (units, type…)
  - Defined as read/write/notify/indicate
- Services
  - Set of related characteristics
  - primary: exposes functionality
  - secondary: referenced by primary
- Profiles
  - Preconfigured global group of services
  - List available from Bluetooth SIG



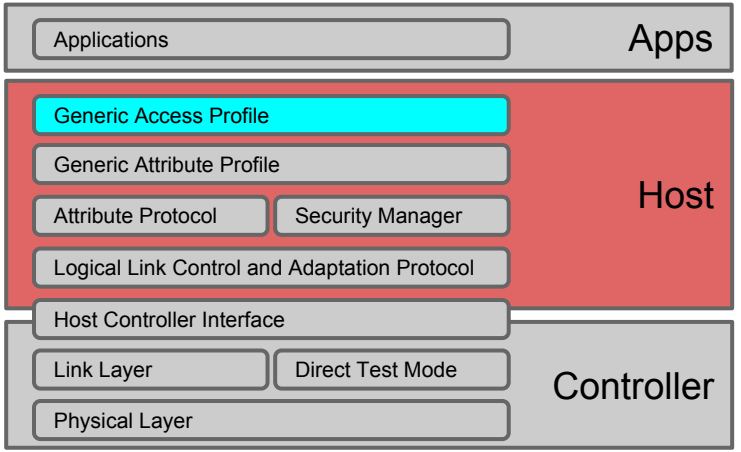
Architecture: ATT and GATT



vancouver [1]

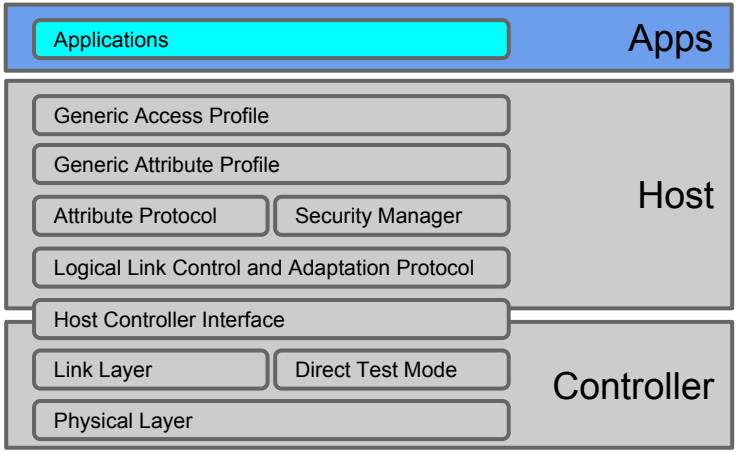
Architecture: Host::Generic Access Profile (GAP)

- Defines procedures
  - discovery of identities, names, capabilities
  - connections
  - security
  - advertising and scan response formats
- Defines roles
  - Broadcaster
    - only advertises
  - Observer
    - receives data from broadcasters
  - Peripheral
    - single connection devices
  - Central
    - device in charge of multiple connections
- Device can only play one roll (BLE 4.0)



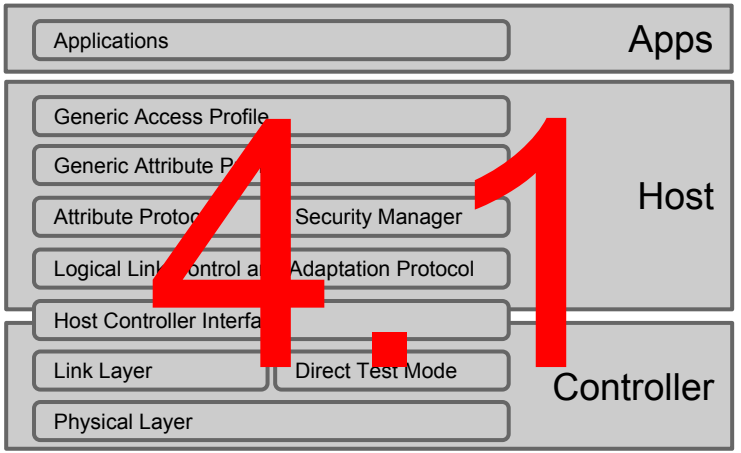
Architecture: Applications

- Applications are built on top
  - Interacts with host layer only
- Different API's depending on the application environment



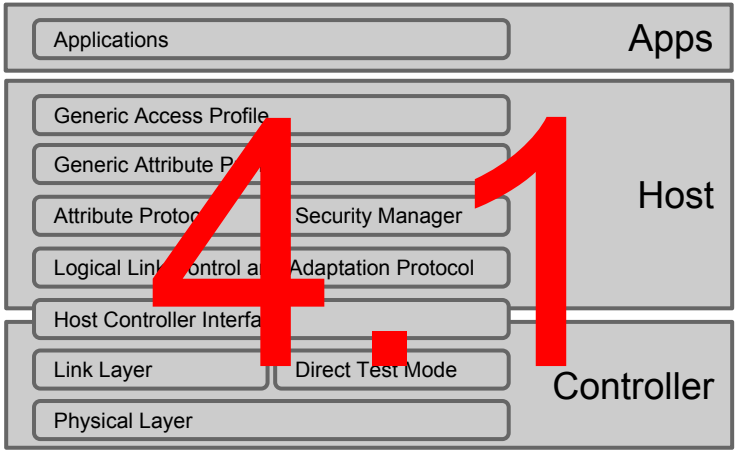
Architecture: But wait...

- BLE 4.1 released Dec 2013
  - changes the world
- Device supports multiple simultaneous roles (e.g. Peripheral and Central)!
- Delay tolerant connections!
- Devices can set up dedicated communication channel!
- Low duty directed advertising!
- Coexistence screening!
- Bulk data transfer!



Architecture: Implications

- Still unclear although...
  - More topologies possible
  - Delay tolerant networks possible
  - BLE to internet directly?
- Many chips do not yet even partially support



Outline

- History of Bluetooth
- Introduction to BLE
- Architecture
  - Controller
  - Host
  - Applications
- Power
- Topology
- Example: Heartbeat Sensor/App
- Competing formats
- Citations

Power

- Power function of lots of things
  - acceptable bit rate error
  - sample rate
  - transmit delays
  - list goes on...
- Minimum transaction time (empty packet) takes approx 3ms
  - at 15mW tx power with 1.5V we get

    15mW / 1.5V = 10mA

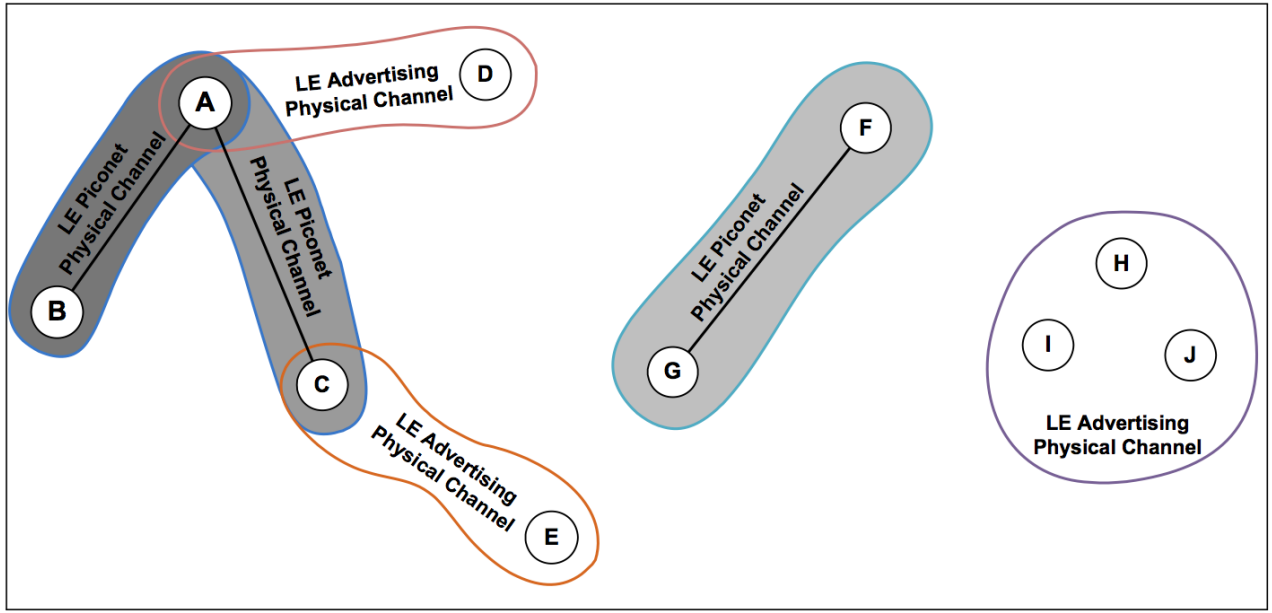
    15mW * .003 S = 45mJ
- 200mAh coin cell -> 200mAh/10mA = 72,000 seconds (20 hours) constant transmit time / 3ms a transaction = 24 million transactions

Outline

- History of Bluetooth
- Introduction to BLE
- Architecture
  - Controller
  - Host
  - Applications
- Power
- Topology
- Example: Heartbeat Sensor/App
- Competing formats
- Citations

Topologies (4.0)

piconets



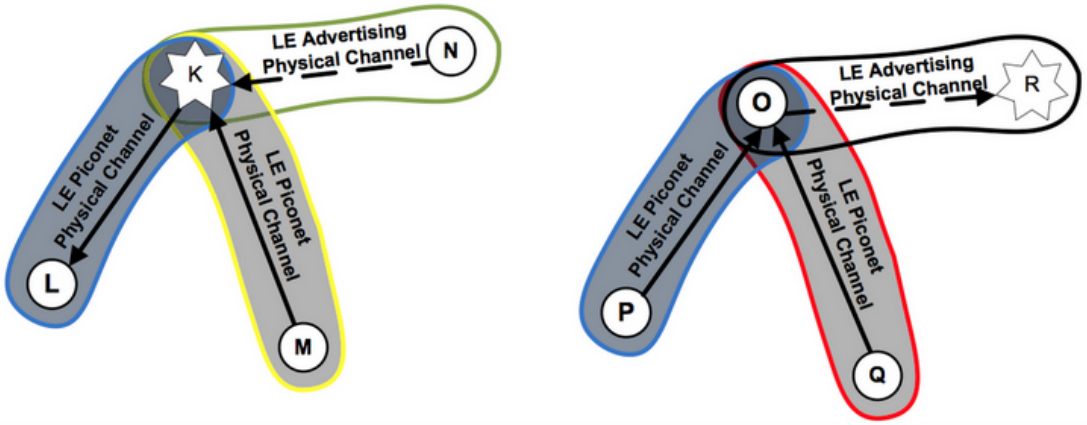
Core [5]

Key:

- Lines indicate a connection
- Groups indicate data transmission

Topologies (4.1)

scatternets



Core [5]

Key:

- Solid arrows point from master to slave
- dashed arrows indicate connection initiation and point from initiator to advertiser
- advertiser are stars

Outline

- History of Bluetooth
- Introduction to BLE
- Architecture
  - Controller
  - Host
  - Applications
- Power
- Topology
- Example: Heartbeat Sensor/App
- Competing formats
- Citations

Example: Heartbeat Sensor / Android Phone

- Simple Heartbeat sensor (HBS) that connects to smartphone over BLE
- HBS will send heartbeat reading and model number
- Devices are initially not connected

Example: Advertisement

- At GAP, HBS set to peripheral and phone is central.
- HBS sends advertising packet at next Advertisement Event
  - Contains services that are supported (raw heartbeat measurement)
    - Gets this info from GATT
- Phone hears and sends Connection Request packet
  - Contains parameters of connection
- HBS receives connection request
- Both devices wait Initial Delay decided by master
- HBS becomes slave and Phone becomes master, starting a piconet.
- Devices are connected!

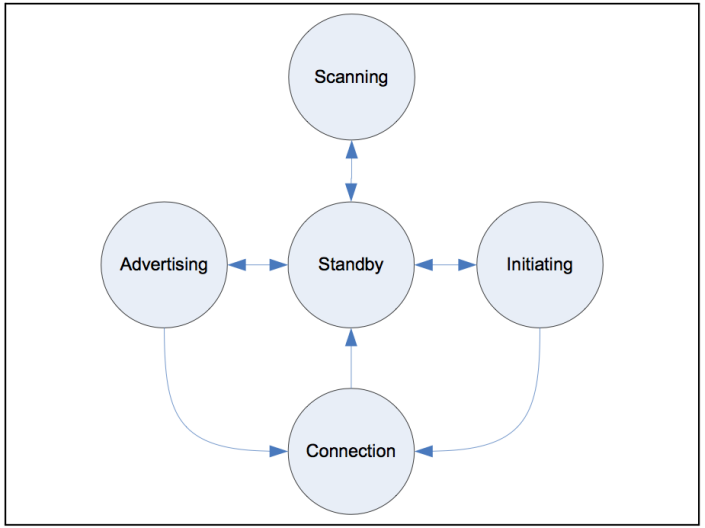
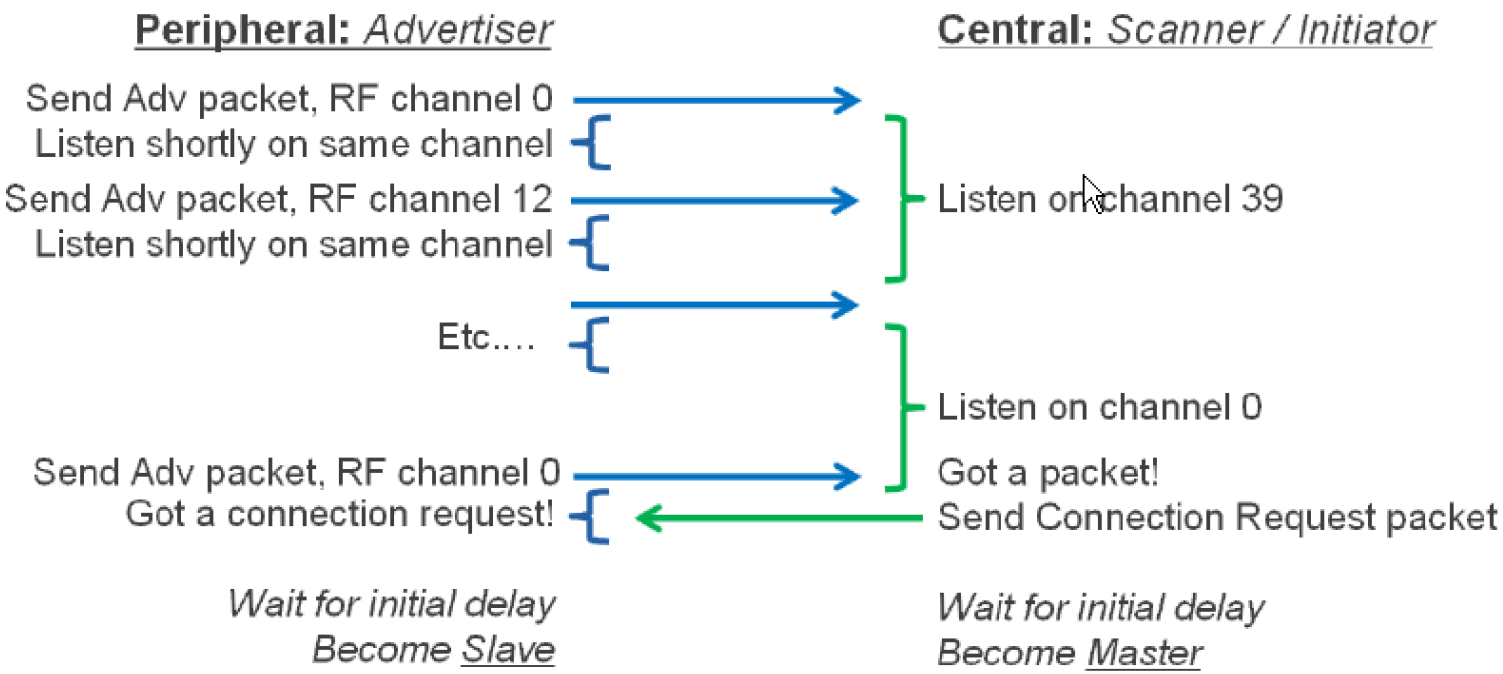


Figure 1.1: State diagram of the Link Layer state machine

Core [5]

Example: Advertisement

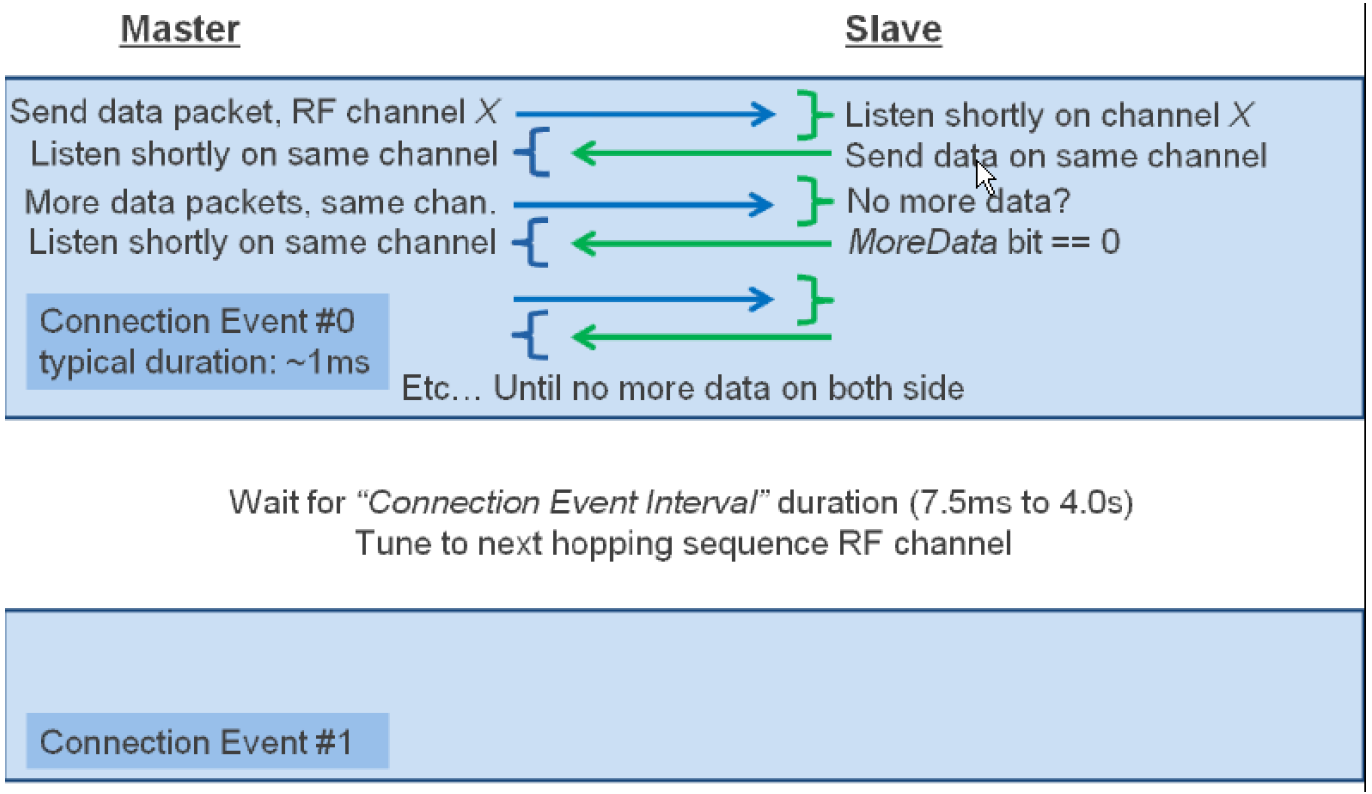


Webinar [4]

Example: Security (optional)

- SMP pairing feature exchange initiated by Master or Slave
- Each calculates a short term key by exchanging random numbers (16byte)
- The short term key is then used to encrypt and decrypt at SMP layer
- A long term key can optionally be transmitted and kept for multiple sessions

Example: Exchanging Data



Webinar [4]

Example: Exchange of Data::Sensor

```
event system_boot(major ,minor ,patch ,build ,ll_version ,protocol_version, hw)
        call gap_set_adv_parameters(1600, 4000, 7)
        call gap_set_mode(2, 2)
        call hardware_set_soft_timer(32768,0,0)
end

event hardware_soft_timer(handle)
        call hardware_adc_read(0,3,0)
end

event hardware_adc_result(input,value)
        call attributes_write(xgatt_hb,0,5,value(0:5))
end

event connection_disconnected(handle,result)
    call gap_set_mode(gap_general_discoverable,gap_undirected_connectable)
end
```

Application.bgs

- writes the value of adc0 to the xgatt_hb attribute

```
xgatt_hb 17
```

ATT.txt

- defines the xgatt_hb attribute

```
<?xml version="1.0" encoding="UTF-8" ?>
<configuration>

    <service uuid="1800">
      <description>Generic Access Profile</description>
      <characteristic uuid="2a00">
        <properties read="true" const="true" />
        <value>Heartbeat Sensor Model</value>
      </characteristic>
      <characteristic uuid="2a01">
        <properties read="true" const="true" />
        <value type="hex">4142</value>
      </characteristic>
    </service>

    <service uuid="deaddead-dead-dead-dead-deaddeaddead" advertise="true">
         <description>Heartbeat Reading Service</description>
         <characteristic uuid="beefbeef-beef-beef-beef-beefbeefbeef" id="xgatt_hb”>
                <properties notify="true" read="true" />
                <description>Raw HB reading</description>
          </characteristic>
    </service>
</configuration>
```

GATT.xml

- *Primary Service:* Information (GAP)
  - UUID: 1800
    - *Characteristic:* Device Name
      - UUID: 2a00
      - readable
    - *Characteristic:* Appearance
      - UUID: 2a01
      - readable
- *Primary Service*: Heartbeat Reading Service
  - UUID: deaddead-dead-dead-dead-deadeadead
    - *Characteristic:* Raw HB reading
      - UUID: beefbeef-beef-beef-beef-beefbeefbeef
      - notification, readable
        - *Attribute*: xgatt_hb

Example: Exchange of Data::Android


```
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothGatt;
import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.BluetoothGattService;
import android.bluetooth.BluetoothProfile;
```

Outline

- History of Bluetooth
- Introduction to BLE
- Architecture
  - Controller
  - Host
  - Applications
- Power
- Topology
- Example: Heartbeat Sensor/App
- Competing formats
- Citations

Competing Formats: Zigbee

- Low cost/power wireless standard
  - Older and more established than bluetooth
- 802.15.4 PHY and MAC
- ISM Band PHY
- Up to 250kbps data transmission
- Star, mesh, and cluster tree topologies
- Slightly higher power than BLE
- Not integrated into widely used devices

(Non)Competing Formats: NFC

- Builds upon RFID standards for two way communication
  - Can be read unpowered
- No universally accepted standard
  - although attempts have been made
- Can be alongside bluetooth and wifi
  - Android Beam uses it to initiate bluetooth pairing
  - S-Beam uses it to initiate wifi direct

Outline

- History of Bluetooth
- Introduction to BLE
- Architecture
  - Controller
  - Host
  - Applications
- Power
- Topology
- Example: Heartbeat Sensor/App
- Competing formats
- Citations

Citations

- **vancouver** http://chapters.comsoc.org/vancouver/BTLER3.pdf
- **ncbi** http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3478807/
- **nordic** http://www.eabeurs.nl/files/7013/7085/2988/3_Introduction_to_Bluetooth_low_energy.pdf
- **webinar** https://developer.bluetooth.org/DevelopmentResources/Pages/Webinars.aspx
- **core** https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=282159
- **world** http://litepoint.com/whitepaper/Bluetooth%20Low%20Energy_WhitePaper.pdf
- **gatt_att** http://teleorigin.com/download/Bluetooth/Low%20Energy/Profile_development_BLE.pdf
- **ee_times** http://www.eetimes.com/document.asp?doc_id=1278966

Questions?

Comments?

Discussion?