Bluetooth

I. Objectives

The objectives of this work are:

- Understand Bluetooth's *Host Controller Interface* (HCI) interface
- Observe the different phases a Bluetooth device goes during its operation
- Identify main Bluetooth protocols and how there are used and behave

II. Duration

This work should be executed in 2h.

III. Procedures

This Work will use:

a) Students' personal PC with Wireshark installed
b) Previously captured traffic exchanges and downloaded from the course available online materials

IV. Network diagram used (approximate)

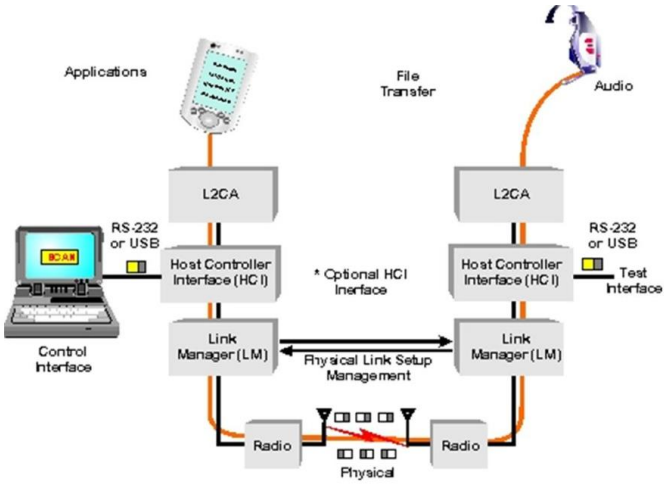


Figure 1: Network diagram used

(http://www.althos.com/tutorial/Bluetooth-tutorial-host-controller-interface-HCI-Layer.html)

V. Used devices

- Master/Controller (left side of the picture)

a) Linux 22.04 PC with TP-Link Archer T5E, AC1200 Wi-Fi Bluetooth 4.2 PCIe Adapter
   - Bluetooth version: 4.2

- Clients/Devices

HP mouse HSA-P007M • Bluetooth version: 4.2	Sony Headphones WF-1000XM4 • Bluetooth version: 5.2 • Bluetooth profiles: A2DP, AVRCP, HFP, HSP • Audio formats: SBC, AAC, LDAC
Sony Headphones WH-1000XM3 • Bluetooth version: 4.2 • Bluetooth profiles: A2DP, AVRCP, HFP, HSP • Audio formats: SBC, AAC, aptX, aptX HD, LDAC	Philips AEA2000/12 adapter • Bluetooth version: 2.1+EDR • Bluetooth profiles: A2DP and AVRCP
RAZER Seiren BT Microphone, RZ19-0415 • Bluetooth Version: 5.0	

A. Scan

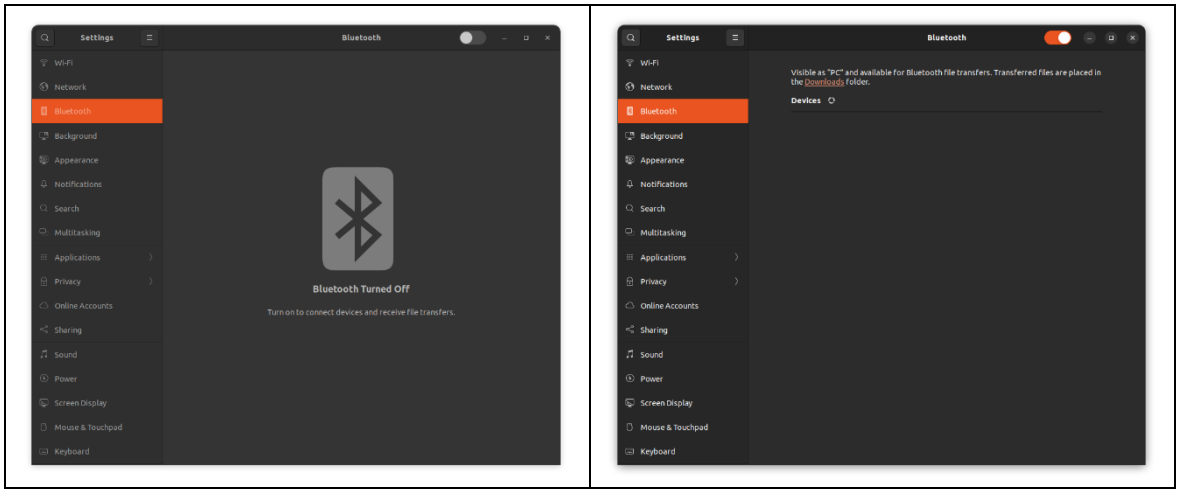
1. Download the Wireshark files, available in eLearning, containing traffic captures taken at an HCI interface for the above listed devices
2. From those, start by opening the capture "**1.PC.BTOn.periodicScanning.pcapng**"

   To better understand this and other captures, go to 'Statistics' → 'Flow Graph' window; when selecting a frame here, it is also selected in Wireshark main window; go to the next frame by pressing 'n'; for the previous one, press 'p'; see each Frame details in Wireshark 'Packed Details' window

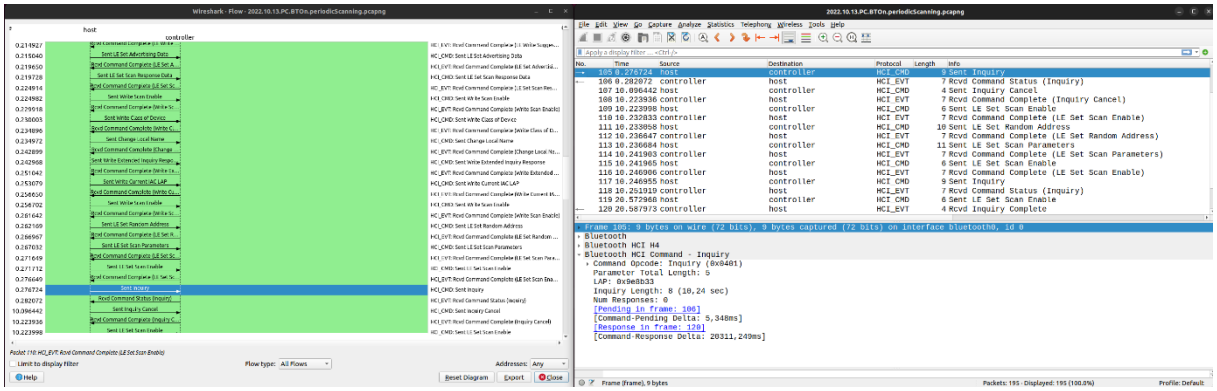
   The following procedures were executed while the capture was running:

   a) The PC Bluetooth interface was turned on Bluetooth settings window; the interface starts scanning immediately

   b) After some seconds of frames capture, it was stopped and saved



3. Go to 'Wireshark' → 'Statistics' → 'Flow Graph' and open that new window side-by-side with the main Wireshark window



4. Order the capture by the column 'Protocol'; scrolling down, identify the protocols present in the capture and the actors involved ('Source' and 'Destination') and the direction of the communication

   R: hci_cmd(sent-host), hci_evt(recieved-controoller)

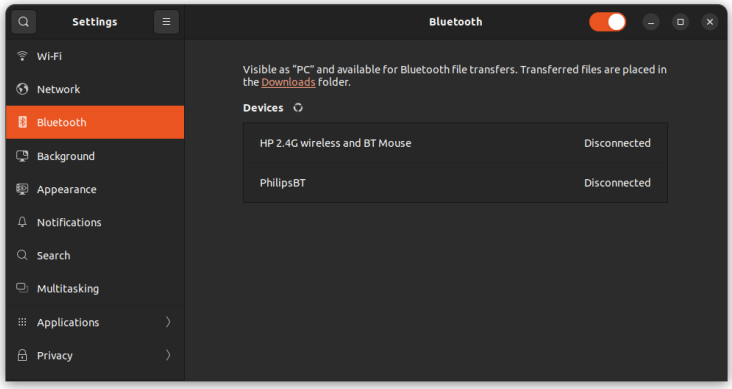
5. Reorder the capture by the column 'No.'; Looking into the 'Source', 'Destination' and 'Protocol' columns, observe the sequence of the messages exchange
6. Observe and analyse the startup process reset, local features, bd addr, class of device,
7. Observe and analyse the periodic Inquiry process. Identify the involved messages of the process.
8. Order the capture using the column 'Info' to see the different messages grouped by its specific type. For instance check the frequency of the Inquiry process. Look at the 'Read', 'Set' and 'Write' messages used in the start-up and the overall process.

B. Pair, connect and use: mouse

9. Now open the capture "**2.HP.Mouse.pair.move.buttons.switchoff.pcapng**"

The following procedures were executed while the capture was running:

a) Turn on Bluetooth on the PC on the Settings Menu



b) Check that no LE Meta (LE Advertising Reports) appear on the capture (no other active devices nearby)

c) Put the device (mouse) in pairing mode

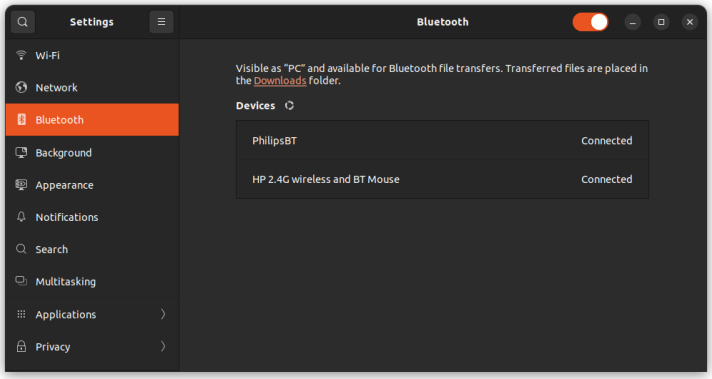
d) Observe in Wireshark that LE Meta (LE Advertising Reports) messages start to appear

e) Order the device to be connected, in the PC's Bluetooth settings menu

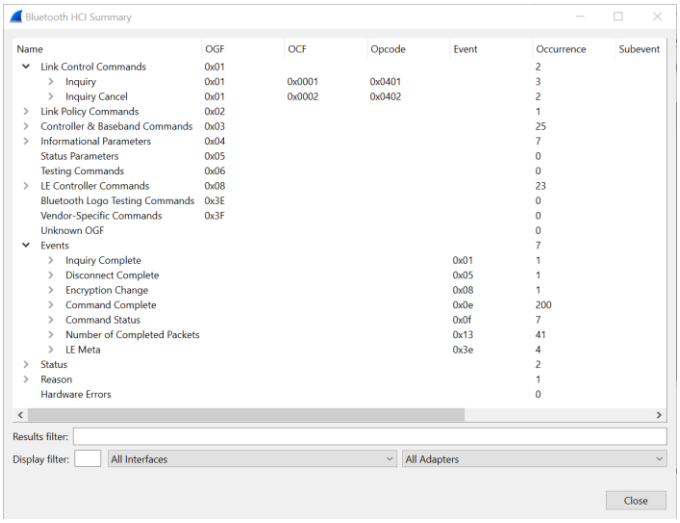
f) Observe that it was successful

g) Press different buttons in the mouse and move it around

h) Stop the capture and saved it



10. Go to 'Wireshark' → 'Wireless' → Bluetooth HCI Summary'; you should get the following window:



a) Open the other vertical groups and analyse the information; keep it open and use the information during the next steps

11. Go to 'Wireshark' → 'Statistics' → 'Flow Graph' and open that new window side-by-side with the main Wireshark window

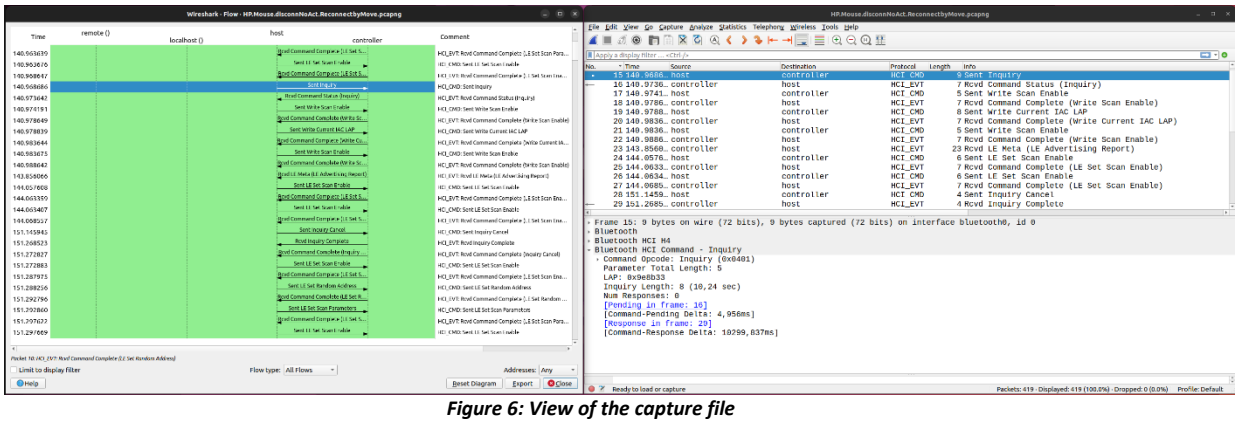


Figure 6: View of the capture file

12. Order capture by the 'Protocol' column and identify the different protocols capture in these interactions.

a) See their different types by applying the visualization filter hci_h4.type==0x{1|2|3|4} (see table at the end); what type of packets is missing? *AAT(atributte protocol) ->layer HOST, HCI, L2CAP, SMP

b) Take note of the entities are the messages exchanged ('Source' and 'Destination')

13. Order again the capture by column 'No.' Observe the pairing and connect procedures that happened after it is requested in the PC Bluetooth Settings window; Identify the following events:

Note: To better analyse the pairing process, order again by 'Protocol' and see the sequence of exchanged messages of that protocol

a) LE Create Connection; register the 'Connection handle' value 571: 0x0e01

b) Pairing Request (*What protocol is used?*) smp

c) Start Encryption 578

14. Now, observe other events; order again by column 'No.'. Look into captured frames 756 to 1298. See the type of frames exchanged during this period.

15. In the details view, check for 'Connection Handle' (ACL), 'Method' (ATT) and 'Value' (ATT) fields. Find the following:

a) Report on battery level 744 -58%

b) Mouse movements (happens from 60sec to 92.4sec of the capture) and Buttons pressed (happens from 78.1sec to 83.9sec of the capture) Rcvd Handle Value Notification Human Interface Device: Report

16. At the end the device is physically switch off, in the built-in button. See what happens at the HCI interface.

Bluetooth standard V4.0 [vol. 2, 7.7.65.2] allows the controller to "queue advertising reports and send information from multiple devices in one LE Advertising Report event".

C. Pair, connect and use, phones

17. Open the capture "**3.0.Sony1000XM3.pairingAndConnect.pcapng**"

    The following procedures were executed while the capture was running:

    a) Turn on Bluetooth on the PC on the Settings Menu

    b) Check that no LE Meta (LE Advertising Reports) appear on the capture (no other active devices nearby)

    c) Put the device (phones) in pairing mode

    d) Observe in Wireshark that LE Meta (LE Advertising Reports) messages start to appear

    e) Order the device to be connected, in the PC's Bluetooth settings menu

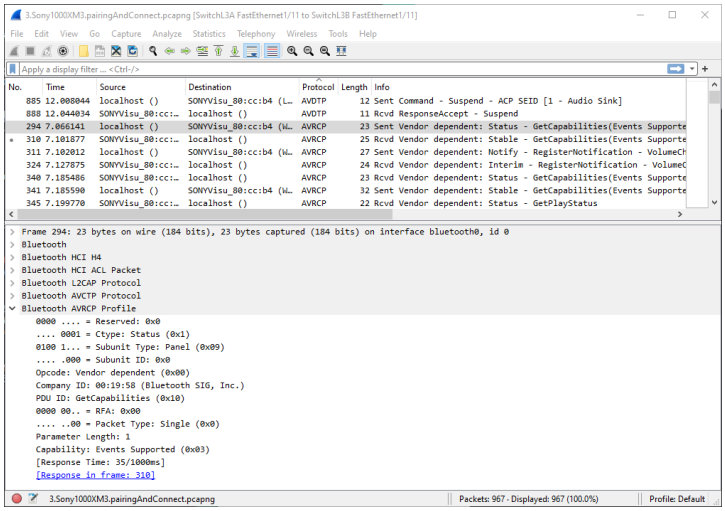
    f) Observe that it was successful

    g) Stop the capture and saved it

18. Order by Protocol and list the protocols involved

    a) By selecting a frame of each protocol, observe the protocol stack in the details window (see example).

    b) Also chech the order there are used



19. Identify the start of the pairing and connect process (it is useful at this step to order again by 'No.')

20. Observe the presence of the protocol RFCOMM; check what parameters are exchanged and the configuration done with it. Also look into HFP.

21. Observe the presence of the protocol AVDTP; check what parameters are exchanged and configuration done with it. Check Frame 217

22. Observe the presence of the protocol AVRPC; check what parameters are exchanged and configuration done with it.

D. Connect and unpair, phones

23. Open the capture "**3.1.Sony1000XM3.powerOn.Connect.unPair.pcapng**"

    The following procedures were executed while the capture was running:

    a) Switch on the device

    b) Wait to see the device switching to Sniff Mode (Frame 745)

    c) In the PC Bluetooth Setting remove the device l2cap -> discon. request

    d) Wait a few seconds, stop the capture and save it

24. Observe the immediate connect requested by the device l2cap: sony and localhost (rcvd)

25. Observe the disconnect process frame 851

E. Audio call (Messenger), phones

26. Open the capture "**4.Sony1000XM3.MessengerCall.pcapng**"; also open 'Statistics → 'Flow Graph'; this capture was obtained in the scope of a Messenger call.

    The following procedures were executed while the capture was running (Phones were already paired and connect):

    a) Started the capture in Wireshark

    b) Started a call in the PC Messenger client to another client

    c) Stop the call in the PC Messenger

    d) Wait to see the last Mode Change

    e) Stop the capture (PC)

hci_sco é entre localhost and sony:sent
e entre remote e local host: received

hci_cmd, evt and hci_sco

27. See Frames 1 to 3; What do they mean? Os clients estavam com os seus dispositivos em modo poupança e sairam do sniff mode.

28. See Frame 4 to 7; check the type of connection being established. Why that? syncronous connection because is voice packets and therefor order and time is important

29. From 8 to 19170, in 29 seconds, audio frames are exchanged bidirectionally.

    a) Apply the visualization filter "hci_h4.type==0x03". hci_SCO -> received/sent DATA

    b) Check on the table at the end of the manual the type of these frame. Local host-> sent
    Sony -> received

    c) Based on that, conclude about frequency and size of the frames

30. Check on of those frames in the 'Packet Details' window. See the protocol stack and 'HCI Packet Type'; see where this exchange fits in the HCI protocol stack present at the end of this manual

31. The call is finished; see what happens in and after Frame 19171. hci_cmd e nao hci_SCO

nao ha l2cap here neste packet nem rfcomm

F. Audio streaming (Spotify), phones

32. Open the capture "**5.Sony1000XM3.Spotify.pcapng**"

    The following procedures were executed while the capture was running (Phones were already paired and connect):

    a) Started the capture in Wireshark

    b) Start streaming on PC (Spotify)

    c) Stop streaming on PC (Spotify)

hci and l2cap

d) Start streaming on phones (Frame 835, 8.62s)

e) Stop streaming on phones (Frame 1380, 23.925s)

f) Stop the capture (PC)

nao ha hci_sco mas sim l2cap (sent) entre local host and remote (connection oriented channel)

33. Identity the type of established connection and see the initial mode change
34. Identity the messages exchange for the audio exchange; observe the direction of the messages
35. Observe what happens when resuming audio stream at the phones.

G. Uni and bidirecctional audio, microfone (with line-out functionality)

36. Open the capture "**6.RaizerMicroph.switchOn.changeToSychroAudio.switchToOff.pcapng**"

The following procedures were executed while the capture was running (the device has been already paired and connected):

a) Start Capture

b) Switch on the device

c) Select the device as Output Device in the Linux Settings window

d) Wait for change mode notification (in Wireshark)

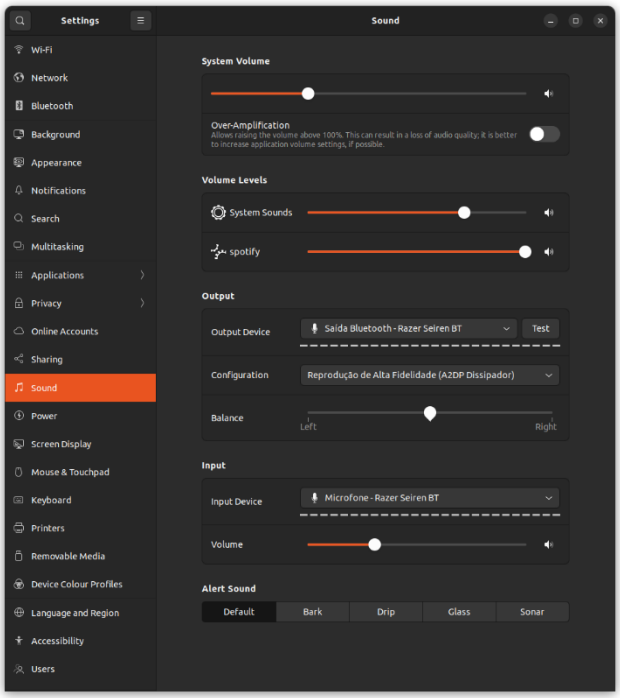
e) Start streaming on Spotify

f) Pause streaming on Spotify

g) Wait for change mode notification (in Wireshark)

h) Resume streaming on Spotify

i) Change audio input device in the Linux Settings window to the Raizen device (see figure)



j) Switch off device

k) Stop the capture

37. Has with previous analysis, identify the involved protocols and their stack
38. Observe the overall process captured in Wireshark and analyse, considering the following guidelines:

a) Initial connection and configuration process between the two devices. It goes from Frame 1 to Frame 181

b) There is an initial exchange of empty audio packets (to be confirmed in following steps) that stop in Frame 899.

c) Frame 901 results from Spotify start.

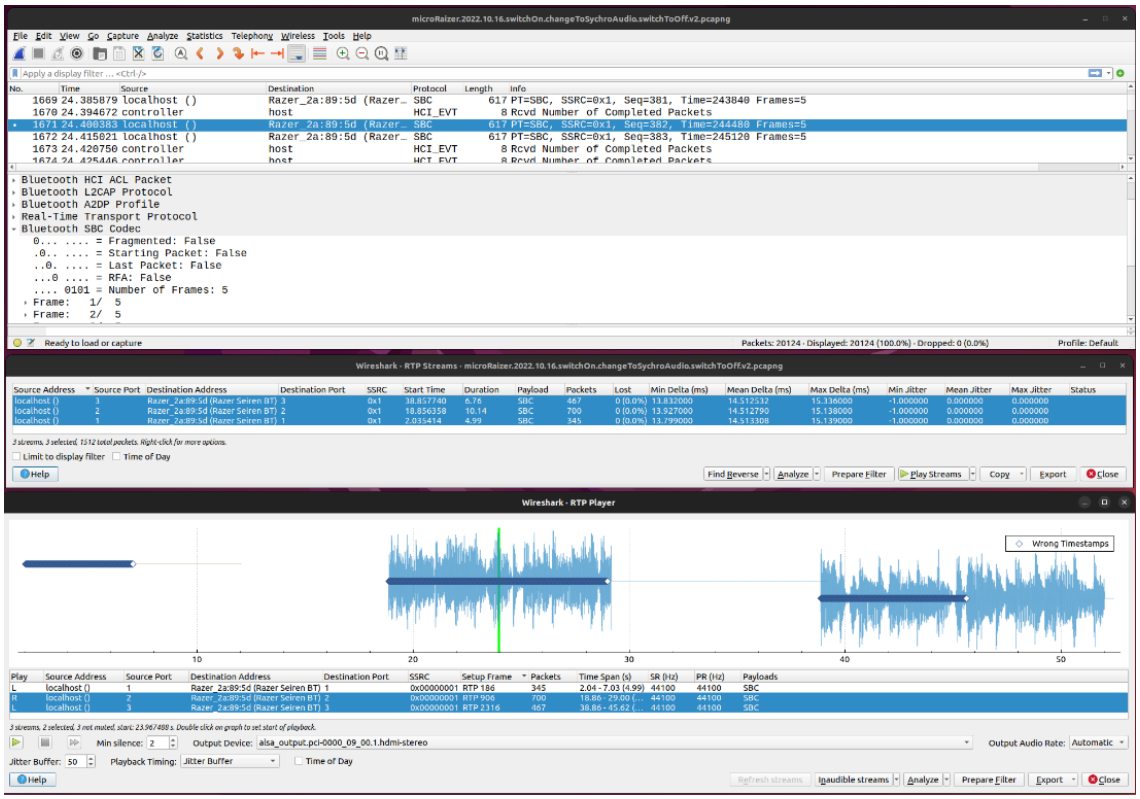
d) Frame 2306 results from Spotify pause.

e) Frame 2311 results from Spotify resuming streaming.

f) Frame 3254 results from the device being added also as audio input device

g) Frame 20088 results from the device being switched off.

39. Go to 'Telephony' --> 'RTP' --> 'RTP Streams' and observed that three RTP were identified. Select then and press 'Play Streams'. With na audio device in your PC you should be able to listen to the 2nd and 3rd streams!



As can be seen in the figure (and in your screen), the first sequence has no audio. The samples with "Wrong timestamps" correspond to audio sent with RTP+SBC (initial setup and unidirectional audio streaming). The last samples correspond the direct, bidirectional audio.

H. Other devices

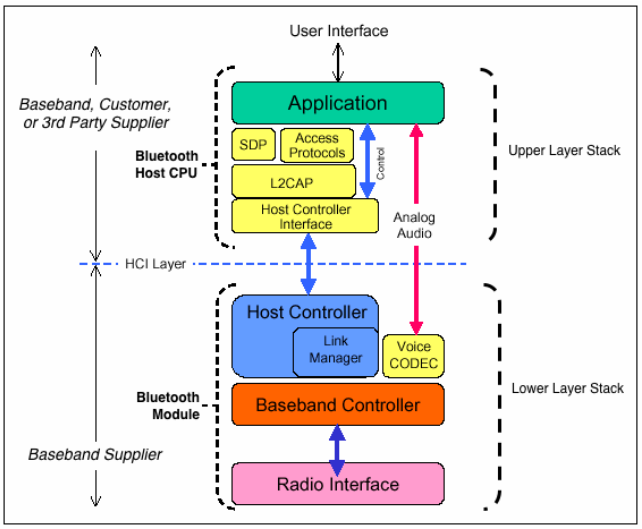
40. Analyse the two other captures:

    7.PhilipsBTaudio.pairing.pcapng and

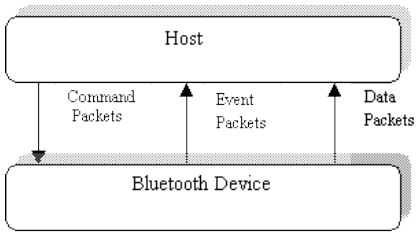
    8.Sony1000XM4.pairing.pcapng

    and do similar analysis to the previous ones.

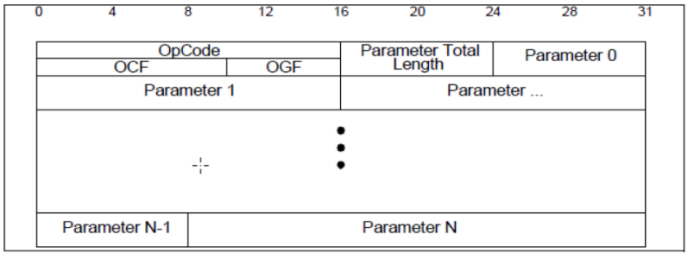
VI. Interface HCI



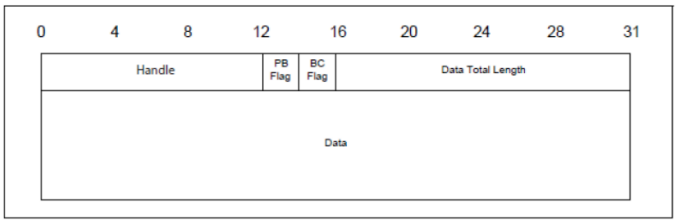
https://hearinghealthmatters.org/wp-content/uploads/sites/9/files/2014/01/BT-Stack.gif



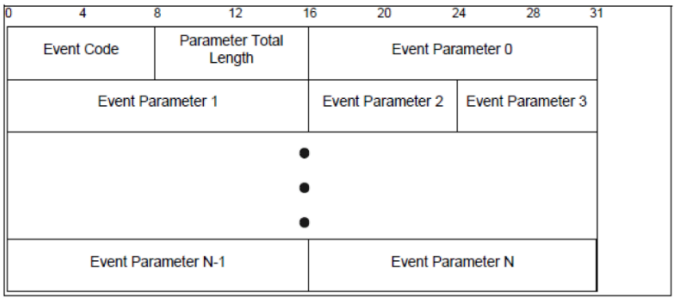
Command Packet



Asynchronous Data Packet



Event Packet



VII. Bluetooth states

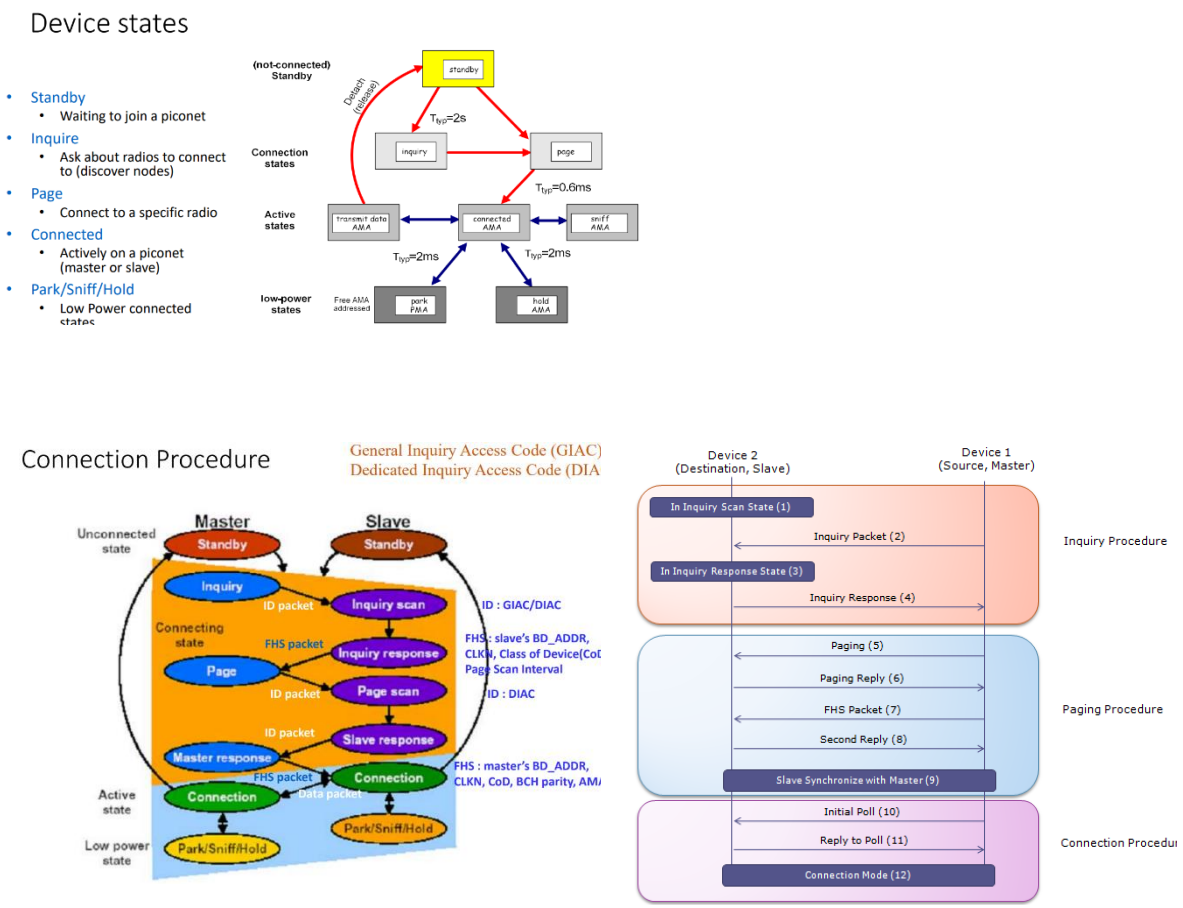
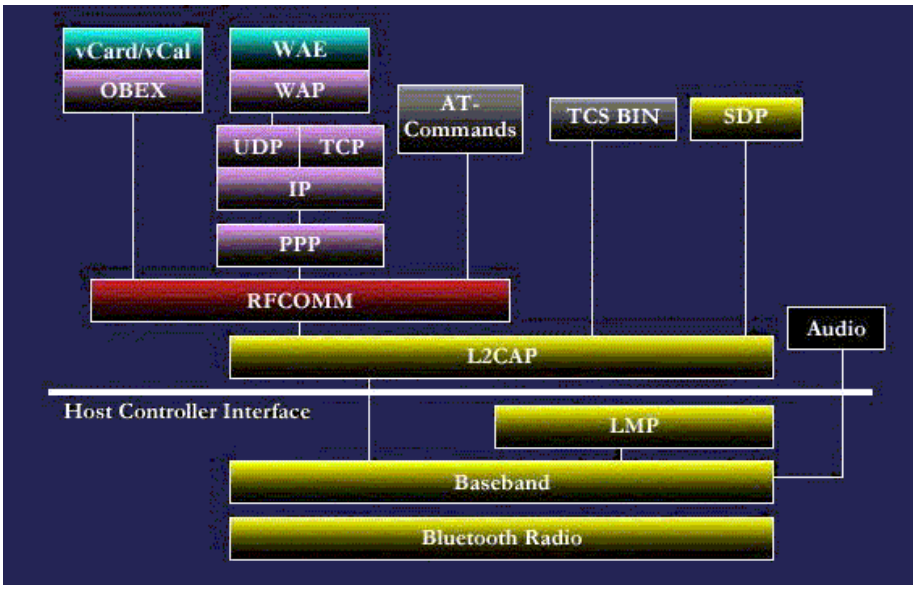


Figure 9: Bluetooth state machine

(http://www.sharetechnote.com/html/Bluetooth_Protocol.html)

VIII. Bluetooth protocols and profiles



http://www.tutorial-reports.com/wireless/bluetooth/protocolstack.php

Attribute Protocol (ATT) is a protocol in the Bluetooth Low Energy (BLE) protocol stack. It defines how data is represented in a BLE server database and the methods by which that data can be read or written.

For example, a fitness tracker gathers data about your steps and heart rate. It acts as a server, holding this data until a smartphone, the client, requests it. This data is stored on the BLE server as attributes.

By definition an attribute is composed of four fields:

attribute type (a universally unique identifier–128-bit number),
attribute handle (a non-zero number for referencing the specific attribute),
attribute permissions (defines if it can be read and/or written) and
attribute value.
These attributes can be read or written using methods that are defined by ATT. For example, requests are sent to servers by the clients and they invoke a response, but notifications are sent to clients by servers and without invoking a response.

IX. Acronyms

NOT Complete. For sure you will find many more on the capture files.

Acronym	Name	Notes
A2DP	Advanced Audio Distribution Profile	
ACP	Acceptor	
ACL	Asynchronous Connection Less	
AVCTP	Audio/Video Control Transport Protocol	Transported in L2CAP
AVDTP	Audio/Video Distribution Transport Protocol	Specifies the **transport protocol** for audio and video **distribution** and streaming Transported in L2CAP
AVRCP	Audio/Video Remote Control Profile	Transported in AVCTP (in L2CAP)
ATT	Attribute Protocol	
DCID	Destination Channel Identifier	
GATT	Generic ATTribute Profile	
HCI	host controller interface	
HFP	Hands-Free Profile	
L2CAP	Logical Link Control and Adaptation Protocol	Supports connection-oriented as well as connectionless services Supports *Synchronous Connection-Oriented* (SCO) links for real-time voice traffic using reserved bandwidth and *Asynchronous Connection-Less* (ACL) links for best-effort traffic
LE	Low Energy	
PSM	Protocol Service Multiplexor	
RFCOMM	Radio Frequency Communication	reliable stream-based protocol providing emulated RS-232 serial ports
RTP	Real-time Transport Protocol	
SBC	Sub-band Coding	Transported in RTP (in
SCID	Source Channel Identifier	
SCO	Synchronous Connection Oriented Link	
SDP	Service Discovery Protocol	
SEID	Stream End-point IDentifier	
SMP	Security Management Protocol	
UIH	Unnumbered Information with Header check	
UUID	Universally Unique IDentifier	

X. Using Wireshark

Preview filters

- hci_h4.direction == 0x00 / 0x01
- hci_h4.type == see table

Packet	Packet Type
Command	0x01
Asynchronous Data	0x02
Synchronous Data	0x03
Event	0x04

- bthci_cmd.opcode == *Command Opcode* (OGF + OCF)
- bthci_cmd.opcode.ocf == *Opcode Command Field*
- bthci_cmd.opcode.ogf == *Opcode Group Field*
- bthci_evt.code == *Event Code*

XI. Useful links

- https://www.bluetooth.com/specifications/specs/
- https://lisha.ufsc.br/teaching/shi/ine5346-2003-1/work/bluetooth/hci_commands.html
- http://oscar.iitb.ac.in/onsiteDocumentsDirectory/Bluetooth/Bluetooth/Help/Host%20Controller%20Interface.htm
- https://gitlab.com/wireshark/wireshark/-/wikis/Bluetooth
- https://software-dl.ti.com/simplelink/esd/simplelink_cc13x2_sdk/1.60.00.29_new/exports/docs/ble5stack/vendor_specific_guide/BLE_Vendor_Specific_HCI_Guide/hci_interface.html
- https://www.wireshark.org/docs/dfref/h/hci_h4.html
- http://www.althos.com/tutorial/Bluetooth-tutorial-title-slide.html