Short Range Wireless Tracker

A Dissertation submitted in the partial fulfillment for the award of MASTER OF TECHNOLOGY

IN

SOFTWARE ENGINEERING

by

Chhavi Deo Shukla

Roll no. 2k12/SWT/16

Under the Essential Guidance of

Dr. Kapil Sharma

Associate Professor

Department of Computer Engineering

Department of Computer Engineering
Delhi Technological University
New Delhi
2012-2015

DECLARATION

I hereby declare that the thesis entitled “**Short Range Mobile Tracker”**
Which is being submitted to the **Delhi Technological University**, in partial fulfillment of the requirements for the award of degree of **Master of Technology in Software Engineering** is an authentic work carried out by me. The material contained in this thesis has not been submitted to any university or institution for the award of any degree.

Chhavi Deo Shukla

Department of Computer Engineering

Delhi Technological University,

Delhi.

CERTIFICATE

DELHI TECHNOLOGICAL UNIVERSITY

(Govt. of National Capital Territory of Delhi)

BAWANA ROAD, DELHI-110042

Date: ____________________

This is to certify that the thesis entitled **"Short Range Wireless Tracker "** submitted by **Chhavi Deo Shukla (Roll Number: 2K12/SWT/16),** in partial fulfillment of the requirements for the award of degree of Master of Technology in Software Engineering, is an authentic work carried out by her under my guidance. The content embodied in this thesis has not been submitted by her earlier to any institution or organization for any degree or diploma to the best of my knowledge and belief.

Project Guide

Dr. Kapil Sharma

Associate Professor

Department of Computer Engineering

Delhi Technological University, Delhi-110042

ACKNOWLEDGEMENT

I take this opportunity to express my deepest gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of this thesis.

Foremost, I would like to express my sincere gratitude to my mentor guide **Dr. Kapil Sharma, Associate Professor, Department of Computer Engineering**, **Delhi Technological University; Delhi** whose benevolent guidance, constant support, encouragement and valuable suggestions throughout the course of my work helped me successfully completes this thesis. Without his continuous support and interest, this thesis would not have been the same as presented here.

Besides my guide, I would like to thank the entire teaching and non-teaching staff in the Department of Computer Science, DTU for all their help during my course of work.

Chhavi Deo Shukla

Table of Contents

LIST OF FIGURES

NOMENCLATURE

APK	**Android Application Package**
BLE	**Bluetooth Low Energy**
BT	**Bluetooth**
EDR	**Enhance Data Rate**
HCI	**Host controller interface**
GPS	**Global positioning System**
SIM	**Subscriber identity Module**
SMS	**Short message Services**
UUID	**Universally unique identifier**

ABSTRACT

Mobile device is getting very popular and becoming utility device. It has been used various purpose as Multimedia device, play station, radio and other processing unit. Once it get lost only tracking is possible using SIM or data connectivity .A new approach to tracking based on short range wireless technology (Bluetooth : Low energy) .

All tracking capable device will be running a client which will be seeking for an advertise message for loss indication on regular basis. Once device detect lost it will start lost server in background. Lost server will advertise loss message on custom blue tooth UUID packet in regular interval. If any tracking capable device nearby receives loss advertise packet will start taking tracking action.

It will create a connection will lost device and get detail of unique info and send information to user register location Once it receive these information of lost device will send an SMS to those number with unique information and its own GPS location in SMS. Helper device own GPS location will help to trace it as lost device will be in 10 meter range.

CHAPTER-1

INTRODUCTION

1.1 **General**

Mobile is getting very popular and becoming utility device. It has been used various purpose as Multimedia device, play station, radio and other processing unit. In case device gets lost then user wants to get intimation of its utility device to trace or get it last presence and usage.

Most of method usage can work if it has on network or data network to check it last usage tracking system.

- Mobile Tracker based on Data networks
- Mobile Tracker based on Sim
- Mobile Tracker Based on Lock device.

Mobile Tracker based on Data Networks: This Tracker system work on data networks and it has cloud base support on the backend .Usage has to register device with his account and it store it unique detail. Once user loss device then it has be login the account mentioned device lost. Based on the device data latch in data network it detect device and give information of device.

Mobile Tracker based on Sim: This Tracker work based program running in device which detect SIM has been changes .Once SIM has changes it start send message to register number for Sim change message notification. In the technique required to latch to network to work.

Mobile Tracker Based on Lock device: In the type of tracker device check weather device has been factory reset .in yes then it ask for credential in not properly given it lock the device to move further. In this case user did not get any information of lost device.

Now question come in case lost device does not comes to network and sim network so no way to get any intimation of that device. Since smart device has many usage as utilities service like multimedia, gaming and other usage.

Here is a new approach to tracking based on short range wireless technology. All tracking capable device will be running a client which will be seeking for an advertise message for loss indication on regular basis.

Once a device has been used for without SIM or Data connectivity (over WIFI or other means), will be treated as a lost possible category and will be prompt to enter a user define credential if not provided correctly will be treated as a lost device.

Device detect lost it will stop client and start lost server in background. Lost server will advertise loss message on 128 bit custom blue tooth UUID packet in regular interval. If any tracking capable device nearby receives loss advertise packet will start taking tracking action.

It will create a connection will lost device and get detail of unique info along with two register number. Once it receive these information of lost device will send an SMS to those number with unique information and its own GPS location in SMS

1.2 Goal of the Thesis:

Goal of this work is to find efficient solution to detect lost device in trivial case devices, and provide less dependent solution on the mobile data and mobile network.
Also solution technology should be common with must have in H/W design i.e. it should be common in the entire device present in market and energy efficient as well.
Taking consideration of above factor we have chosen wireless technology and in that Bluetooth most commonly used and essentially find in most of H/W in surrounding.
So Bluetooth has been shortlist for implementation.

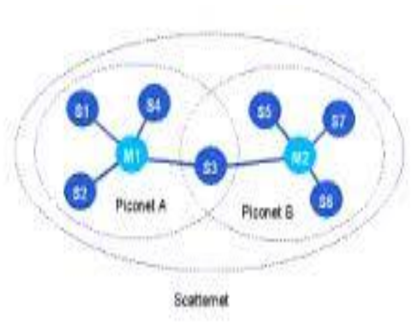
CHAPTER-2

BLUETOOTH RADIO SYSTEM ARCHITECTURE

Bluetooth devices operate at 2.4GHz, in the globally available, license-free, ISM band. That is the bandwidth reserved for general use by Industrial, Scientific and Medical applications worldwide. Since this radio band is free to be used by any radio transmitter as long as it satisfies the regulations, the intensity and the nature of interference can't be predicted. Therefore, the interference immunity is very important issue for Bluetooth. Generally, interference immunity can be obtained by interference suppression or avoidance. Suppression can be obtained by coding or direct-sequence spreading, but the dynamic range of interfering signals in ad hoc networks can be huge, so practically attained coding and processing gains are usually inadequate. Avoidance in frequency is more practical. Since ISM band provides about 80MHz of bandwidth and all radio systems are band limited, there is a high probability that a part of the spectrum can be found without a strong interference.

Considering all this, FH-CDMA (Frequency Hopping - Code Division Multiple Access) technique has been chosen to implement the multiple access scheme for the Bluetooth. It combines a number of properties, which make it the best choice for an ad hoc radio system. It fulfills the spreading requirements set in the ISM band, i.e. on average the signal can be spread over a large frequency range, but instantaneously only a small part of the bandwidth is occupied, avoiding most of potential interference. It also doesn't require neither strict time synchronization (like TDMA), nor coordinated power control (like DS-CDMA). In the 2.45GHz ISM band, a set of 79 hop carriers has been defined, at 1MHz spacing. A nominal hop dwell time is 625 us. Full-duplex communication is achieved by applying time-division duplex (TDD), and since transmission and reception take place at different time slots, they also take place at different hop carriers. A large number of pseudo-random hopping sequences have been defined, and the particular sequence is determined by the unit that controls the FH channel. That unit is usually called the *master* and it also defines timing parameters during the certain session. All other devices involved in the session, the *slaves*, have to adjust their spreading sequences and clocks to the master's.

An FH Bluetooth channel is associated with the piconet. As mentioned earlier, the master unit defines the piconet channel by providing the hop sequence and the hop phase. All other units participating in the piconet are slaves. However, since the Bluetooth is based on peer communications, the master/slave role is only attributed to a unit for the duration of the piconet. When the piconet is cancelled, the master and slaves roles are canceled too. In addition to defining the piconet, the master also controls the traffic on the piconet and takes care of access control. The time slots are alternatively used for master and slaves transmission. In order to prevent collisions on the channel due to multiple slave transmissions, the master applies a polling technique, for each slave-to-master slot the master decides which slave is allowed to transmit. If the master has no information to send, it still has to poll the slave explicitly with a short poll packet. This master control effectively prevents collisions between the participants in the piconet, but independent collocated piconets may interfere with one another when they occasionally use the same hop carrier. This can happen because units don't check for a clear carrier (no listen-before-talk). If the collision occurs, data are retransmitted at the next transmission opportunity. Due to the short dwell time, collision avoidance schemes are less appropriate for FH system.



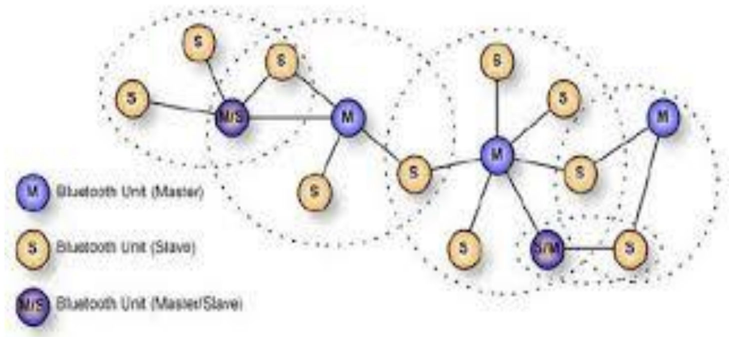


FIGURE1: Bluetooth Scatter-net [Ref. www.bluetooth.org]

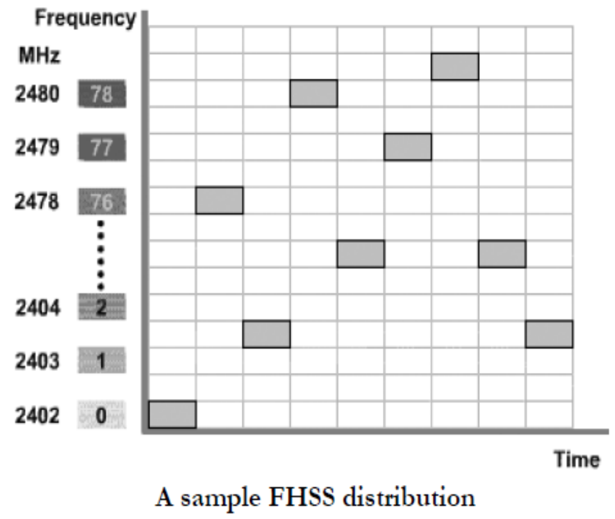


FIGURE 2: BLUETOOTH FHSS DISTRIBUTION
[Ref: GAP Profile Specfcation-www.bluetooth.org]

CHAPTER-3

BLUETOOTH EVOLUTION

Bluetooth v1.0 and v1.0B

Versions 1.0 and 1.0B had many problems, and manufacturers had difficulty making their products interoperable. Versions 1.0 and 1.0B also included mandatory Bluetooth h ardware device address (BD_ADDR) transmission in the Connecting process (rendering anonymity impossible at the protocol level), which was a major setback for certain services planned for use in Bluetooth environments.

Bluetooth v1.1

- Ratified as IEEE Standard 802.15.1-2002
- Many errors found in the 1.0B specifications were fixed.
- Added possibility of non-encrypted channels.
- Received Signal Strength Indicator (RSSI).

Bluetooth v1.2

This version is backward compatible with 1.1 and the major enhancements include the following:

- Faster Connection and Discovery
- *Adaptive frequency-hopping spread spectrum (AFH)*, which improves resistance to radio frequency interference by avoiding the use of crowded frequencies in the hopping sequence.
- Higher transmission speeds in practice, up to 721 kbit/s, than in v1.1.
- Extended Synchronous Connections (eSCO), which improve voice quality of audio links by allowing retransmissions of corrupted packets, and may optionally increase audio latency to provide better concurrent data transfer.
- Host Controller Interface (HCI) operation with three-wire UART.
- Ratified as IEEE Standard 802.15.1-2005
- Introduced Flow Control and Retransmission Modes for L2CAP.

Bluetooth v2.0 + EDR

This version of the Bluetooth Core Specification was released in 2004 and is backward compatible with the previous version 1.2. The main difference is the introduction of an Enhanced Data Rate (EDR) for faster data transfer. The nominal rate of EDR is about 3 Mbit/s, although the practical data transfer rate is 2.1 Mbit/s. EDR uses a combination of GFSK and Phase Shift Keyingmodulation (PSK) with two variants, π/4-DQPSK and 8DPSK. EDR can provide a lower power consumption through a reduced duty cycle. The specification is published as "Bluetooth v2.0 + EDR" which implies that EDR is an optional feature. Aside from EDR, there are other minor improvements to the 2.0 specification, and products may claim compliance to "Bluetooth v2.0″ without supporting the higher data rate.

Bluetooth v2.1 + EDR

Bluetooth Core Specification Version 2.1 + EDR is fully backward compatible with 1.2, and was adopted by the Bluetooth SIG on July 26, 2007.
The headline feature of 2.1 is secure simple pairing (SSP): this improves the pairing experience for Bluetooth devices, while increasing the use and strength of security.
2.1 allows various other improvements, including "Extended inquiry response" (EIR), which provides more information during the inquiry procedure to allow better filtering of devices before connection; and sniff subrating, which reduces the power consumption in low-power mode.

Bluetooth v3.0 + HS

Version 3.0 + HS of the Bluetooth Core Specification was adopted by the Bluetooth SIG on April 21, 2009. Bluetooth 3.0+HS provides theoretical data transfer speeds of up to **24 Mbit/s,**though not over the Bluetooth link itself. Instead, the Bluetooth link is used for negotiation and establishment, and the high data rate traffic is carried over a collocated 802.11 link.
The main new feature is AMP (Alternate MAC/PHY), the addition of 802.11 as a high speed transport. The High-Speed part of the specification is not mandatory, and hence only devices sporting the "+HS" will actually support the Bluetooth over 802.11 high-speed data transfer. A Bluetooth 3.0 device without the "+HS" suffix will not support High Speed, and needs to only support a feature introduced in Core Specification Version 3.0 or earlier Core Specification Addendum 1.

L2CAP Enhanced modes

Enhanced Retransmission Mode (ERTM) implements reliable L2CAP channel, while Streaming Mode (SM) implements unreliable channel with no retransmission or flow control. Introduced in Core Specification Addendum 1.

Alternate MAC/PHY
Enables the use of alternative MAC and PHYs for transporting Bluetooth profile data. The Bluetooth radio is still used for device discovery, initial connection and profile configuration, however when large quantities of data need to be sent, the high speed alternate MAC PHY 802.11 (typically associated with Wi-Fi) will be used to transport the data. This means that the proven low power connection models of Bluetooth are used when the system is idle, and the faster radio is used when large quantities of data need to be sent. AMP links require enhanced L2CAP modes.

Enhanced Power Control
Updates the power control feature to remove the open loop power control, and also to clarify ambiguities in power control introduced by the new modulation schemes added for EDR. Enhanced power control removes the ambiguities by specifying the behaviour that is expected. The feature also adds closed loop power control, meaning RSSI filtering can start as the response is received. Additionally, a “go straight to maximum power” request has been introduced. This is expected to deal with the headset link loss issue typically observed when a user puts their phone into a pocket on the opposite side to the headset.

Bluetooth 4.0 (BLE) Bluetooth low energy
This version's development

- Ease of implementation and multi-vendor interoperability
- Ultra-low peak, average and idle mode power consumption
- Low cost of integration
- Power handling
- Resistance to interference

Bluetooth Low Energy technology extends the personal area network (PAN) to include Bluetooth enabled devices that are powered by small, coin-cell batteries. With low energy technology, sports and health care equipment, human interface (HIDs) and entertainment devices are enhanced. The technology can be built into products such as watches, wireless keyboards, gaming and sports sensors, which can then connect and communicate with host devices, such as mobile phones and personal computers.

Bluetooth 4.1 (Still to be commercialized)

1. **Coexistence**

Bluetooth and 4G (LTE) famously don’t get on: their signals interfere degrading one another’s performance and draining battery life. Bluetooth 4.1 eliminates this by coordinating its radio with 4G automatically so there is no overlap and both can perform at their maximum potential. Given most phones will come with 4G next year this is a vital improvement.

2. Smart connectivity

Rather than carry a fixed timeout period, Bluetooth 4.1 will allow manufacturers to specify the reconnection timeout intervals for their devices. This means devices can better manage their power and that of the device they are paired to by automatically powering up and down based on a bespoke power plan. In short: devices are no longer all treated the same and the randomness of Bluetooth connections / disconnections and the power drain this causes should reduce dramatically.

3. Improved Data Transfer

Bluetooth 4.1 devices can act as both hub and end point simultaneously. This is hugely significant because it allows the host device to be cut out of the equation and for peripherals to communicate independently. For example, whereas previously a smartwatch would need to talk to your phone to get data from a heart monitor, now the smartwatch and heart monitor can talk directly saving your phone’s battery and then upload their compiled results directly to your phone.

CHAPTER-4

BLUETOOTH STACK

The Bluetooth protocol stack is defined as a series of layers, though there are some features which cross several layers. A Bluetooth device can be made up of two parts: a host implementing the higher layers of the protocol stack, and a module implementing the lower layers. This separation of the layers can be useful for several reasons. For example, hosts such as PCs have spare capacity to handle higher layers, allowing the Bluetooth device to have less memory and a less powerful processor, which leads to cost reduction. Also, the host device can sleep and be awoken by an incoming Bluetooth connection. Of course, an interface is needed between the higher and lower layers, and for that purpose the Bluetooth defines the Host Controller Interface (HCI). But for some small and simple systems, it is still possible to have all layers of the protocol stack run on one processor. An example of such a system is a headset.

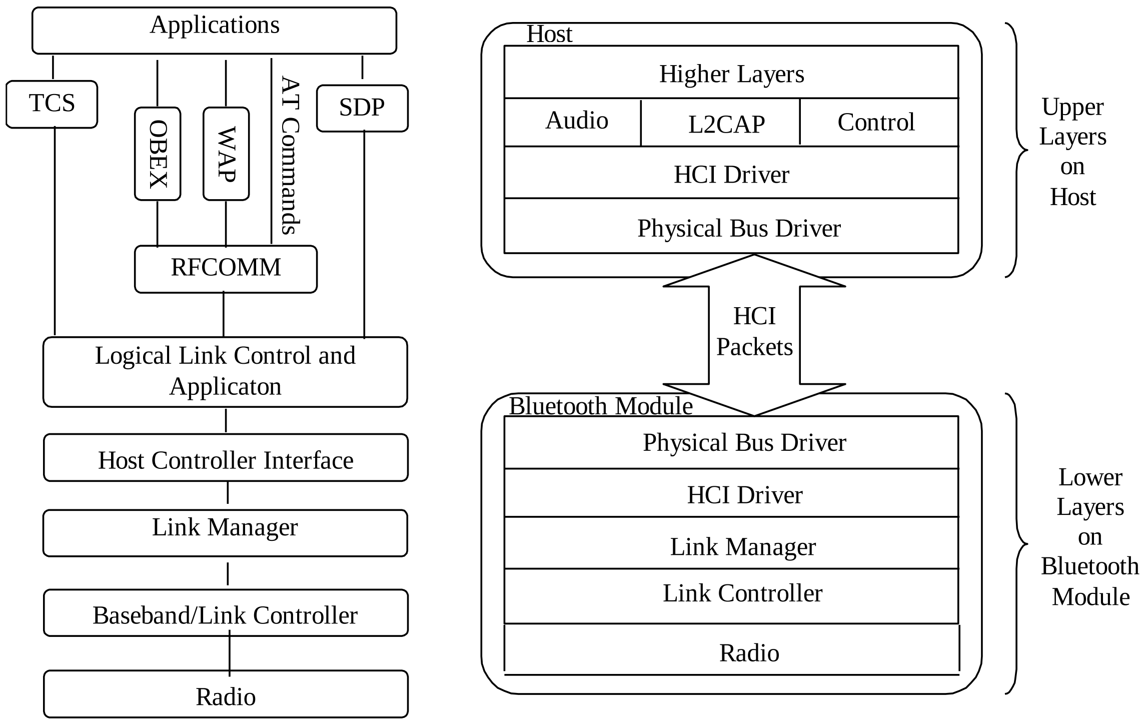


Figure 3: Bluetooth Protocol Stack
[Ref: GAP Profile Specification-www.bluetooth.org]

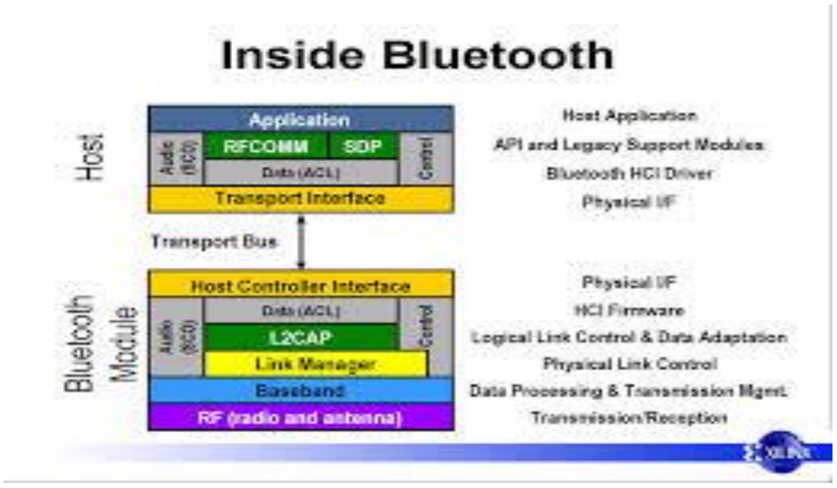


Figure 4: Bluetooth System
[Ref: GAP Profile Specifcation-www.bluetooth.org]

4.1 Bluetooth Controller

Baseband - There are two basic types of physical links that can be established between a master and a slave:

- Synchronous Connection Oriented (SCO)
- Asynchronous Connection-Less (ACL)

An SCO link provides a symmetric link between the master and the slave, with regular periodic exchange of data in the form of reserved slots. Thus, the SCO link provides a circuit-switched connection where data are regularly exchanged, and as such it is intended for use with time-bounded information as audio. A master can support up to three SCO links to the same or to different slaves. A slave can support up to three SCO links from the same master.

An ACI link is a point-to-multipoint link between the master and all the slaves on the piconet. It can use all of the remaining slots on the channel not used for SCO links. The ACL link provides a packet-switched connection where data are exchanged sporadically, as they become available from higher layers of the stack. The traffic over the ACL link is completely scheduled by the master.

Each Bluetooth device has a 48 bit IEEE MAC address that is used for the derivation of the *access code*. The access code has pseudo-random properties and includes the identity of the piconet master. All the packets exchanged on the channel are identified by this master identity. That prevents packets sent in one piconet to be falsely accepted by devices in another piconet that happens to use the same hopping frequency in the certain time slot. . All packets have the same format, starting with an access code, followed by a packet header and ending with the user payload.

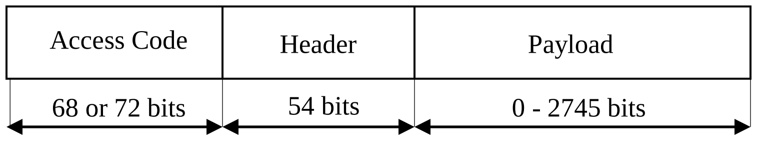


Figure 5**:** Bluetooth packet structure
[Ref: GAP Profile Specification-www.bluetooth.org]

The access code is used to address the packet to a specific device. The header contains all the control information associated with the packet and the link. The payload contains the actual message information. The Bluetooth packets can be 1, 3, or 5 slots long, but the multislot packets are always sent on a single-hop carrier.

The Link Controller - The link control layer is responsible for managing device discoverability, establishing connections and maintaining them. In Bluetooth, three elements have been defined to support connection establishment: scan, page and inquiry.

Inquiry is a process in which a device attempts to discover all the Bluetooth enabled devices in its local area. A unit that wants to make a connection broadcasts an inquiry message that induces the recipients to return their addresses. Units that receive the inquiry message return an FHS (FH-synchronization) packet which includes, among other things, their identity and clock information. The identity of the recipient is required to determine the page message and wake-up sequence. For the return of FHS packets, a random backoff mechanism is used to prevent collisions.

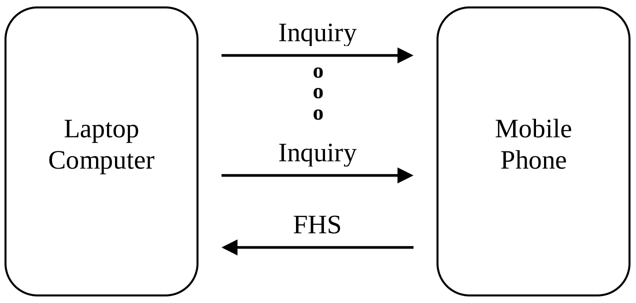


Figure 6: Discovering a Bluetooth device
[Ref: GAP Profile Specification-www.bluetooth.org]

A unit in idle mode wants to sleep most of the time to save power, but, from time to time, it also has to listen whether other units want to connect (page scan). In truly ad hoc system, there is no common control channel a unit can lock to in order to listen for page messages. So, every time the unit wakes up, it scans at a different hop carrier for an extended time. A trade-off has to be made between idle mode power consumption and response time: increasing the sleep time reduces power consumption but prolongs the time before an access can be made. The unit that wants to connect has to solve the frequency-time uncertainty: it doesn't know when the idle unit will wake up and on which frequency. For that reason, the paging unit transmits the access code repeatedly at different frequencies: every 1.25ms the paging unit transmits two access codes and listens twice for a response. In 10ms period, 16 different hop carriers are visited. If the idle unit wakes up in any of these 16 frequencies, it will receive the access code and start with a connection setup procedure. First, it will notify the paging unit by returning a message, and then it will transmit a FHS packet which contains all of the pager's information. This information is then used by both units to establish the piconet. Once a baseband link is established, the master and slave can exchange roles if they wish, so that slave becomes master and master becomes slave.

It should be noted that the control of links rests completely with the local device. If it doesn't make itself discoverable by page scanning it cannot be found, if it does not make itself connectable by page scanning it cannot be linked with, and once in a connection it is free to disconnect without warning at any time.

Audio - Audio data is carried via SCO (Synchronous Connection Oriented) channels. These SCO channels use pre-reserved slots to maintain temporal consistency of the audio carried on them. This allows us to build devices such as wireless headsets, microphones and headphones using Bluetooth for many consumer products such as cellular phones, call centre switchboards, or even personal musical playback.

There are two routes for audio to pass through a Bluetooth system: through the HCI as data in HCI packets, and via direct PCM connection to the baseband CODECs.

The HCI route has some deficiencies in carrying audio data, i.e. packets crossing the HCL are subject to flow control and therefore to variable latency due to microcontroller executing the HCI and LM (Link Manager) tasks. The direct PCM route is not well specified in the Bluetooth specifications, but is very common in commercial implementations.

The Link Manager - The host drives a Bluetooth device through Host Controller Interface (HCI) commands, but it is the link manager that translates those commands into operations at the baseband level. Its main functions are to control piconet management (establishing and destruction of the links and role change), link configuration, and security and QoS functions.

Link manager communicates with its peers on other devices using the Link Management Protocol (LMP). Every LMP message begins with a flag bit which is 0 if a master initiated the transaction and 1 if the slave initiated the transaction. That bit is followed by a 7-bit Operation Code, and by the message's parameters.

Then a link is first set up, it uses single-slot packets by default. Multi-slot packets make more efficient use of the band, but there are some occasions when they can't be used, for example on noisy links or if SCO links don't leave sufficient space between their slots for multi-slot packets.

LMP also provides a mechanism for negotiating encryption modes and coordinating encryption keys used by devices on both ends of the link. In addition, LMP supports messages for configuration of the quality of service on a connection. Packet types can automatically change according to the channel quality, so that the data can be transferred at a higher rate when the channel quality is good, and on lower rates with more error protection if the channel quality deteriorates.

4.2 Bluetooth Host

Logical Link Control and Adaptation Protocol (L2CAP) - Logical Link Control and Adaptation Protocol takes data from higher layers of the Bluetooth stack and from applications and sends them over the lower layers of the stack. It passes packets either to the HCI, or in a host-less system directly to the Link Manager. The major functions of the L2CAP are:

- Multiplexing between different higher layer protocols to allow several higher layer links to share a single ACL connection. L2CAP uses channel numbers to label packets so that, when they are received, they can be routed to the correct place.
- Segmentation and reassembly to allow transfer of larger packets than lower layers support.
- Quality of service management for higher layer protocols.

All applications must use L2CAP to send data. It is also used by Bluetooth's higher layers such as RFCOMM and SDP, so L2CAP is a compulsory part of every Bluetooth system.

RFCOMM - RFCOMM is a simple, reliable transport protocol that provides emulation of the serial cable line settings and status of an RS-232 serial port. It provides connections to multiple devices by relying on L2CAP to handle multiplexing over single connection. RFCOMM supports two types of devices:

- Type 1 - Internal emulated serial port. These devices usually are the end of a communication path, for example a PC or printer.
- Type 2 - Intermediate device with physical serial port. These are devices that sit in the middle of a communication path, for example a modem.

Up to 30 data channels can be set up, so RFCOMM can theoretically support 30 different services at once. RFCOMM is based on GSM TS 07.10 standard, which is an asymmetric protocol used by GSM cellular phones to multiplex several streams of data onto one physical serial cable.

The Service Discovery Protocol - One of the most important members of the Bluetooth protocol stack is Service Discovery Protocol (SDP). It provides a means for an SDP client to access information about services offered by SDP servers. An SDP server is any Bluetooth device which offers services to other Bluetooth devices. Information about services is maintained in SDP databases. There is no centralized database, so each SDP server maintains its own database. The SDP database is simply a set of records describing all the services which a Bluetooth device can offer to another Bluetooth device, and service discovery protocol provides a means for another device to look at these records. To browse service classes, or get information about a specific service, SDP clients and servers exchange messages which are carried in SDP Protocol Data Units (PDUs). The first byte of PDU is an ID, identifying the message in the PDU. Services have Universally Unique Identifiers (UUIDs) that describe them. The services defined by the Bluetooth profiles have UUIDs assigned by the standard, but service providers can define their own services and assign their own UUIDs to those services.

SDP relies on L2CAP links being established between SDP client and server, before retrieving SDP information. Stages in setting up an SDP connection are

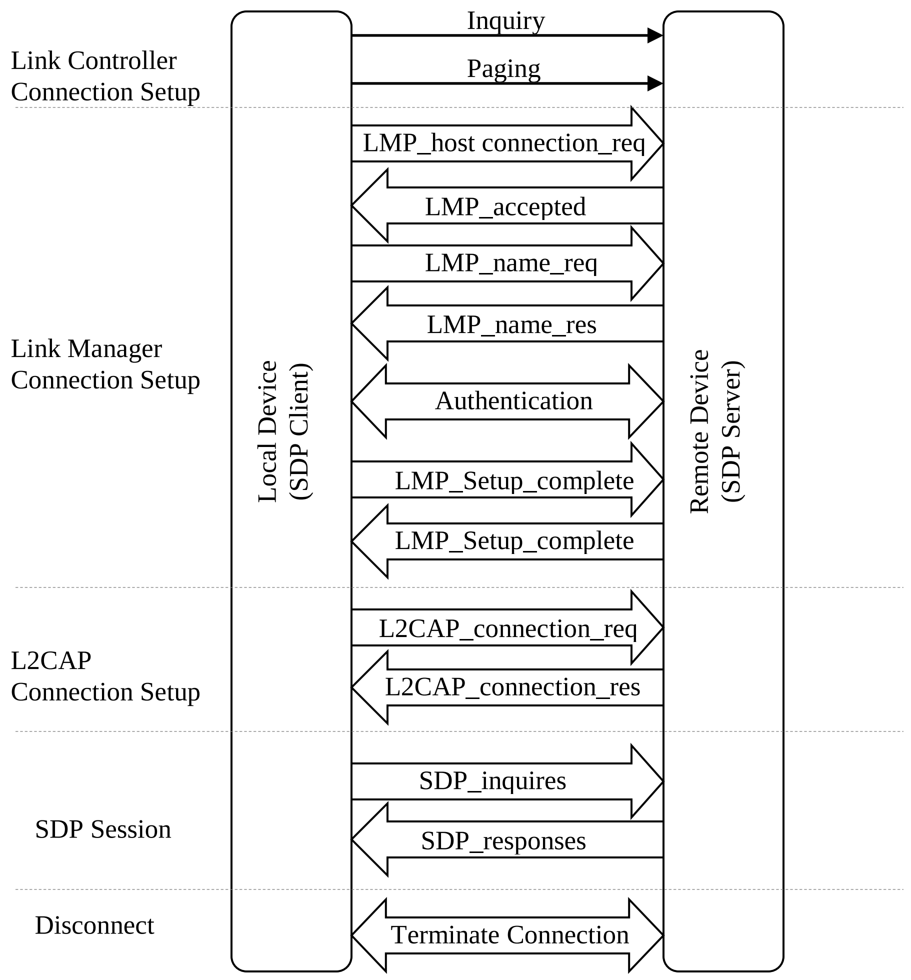


Figure 7: Stages in setting up an SDP session
[Ref: SDP Profile Specification-www.bluetooth.org]

CHAPTER-5

PROTOCOLS AND PROFILES

5.1 BLUETOOTH PROTOCOLS:

As mentioned at the begin, one of the most important characteristics of the Bluetooth specification is that it should allow devices from lots of different manufacturers to work with one another. For that reason, Bluetooth is designed in such a way to allow many different protocols to be run on top of it. Some of these protocols are:

***The Wireless Access Protocol (WAP)* -** WAP provides a protocol stack similar to the IP stack, but it is tailored for the needs of mobile devices. It supports the limited display size and resolution typically found on mobile devices by providing special formats for Web pages which suit their capabilities. It also provides for the low bandwidth of mobile devices by defining a method for WAP content to be compressed before it is transmitted across a wireless link. WAP can use Bluetooth as a bearer layer in the same way as it can use GSM, CDMA and other wireless services. The WAP stack is joined to the Bluetooth stack using User Datagram Protocol (UDP), Internet Protocol (IP) and Point to Point Protocol (PPP).

Object Exchange Protocol (OBEX) - OBEX is a protocol designed to allow a variety of devices to exchange data simply and spontaneously. Bluetooth has adopted this protocol from the Infrared Data Association (IrDA) specifications. OBEX has a client/server architecture and allows a client to push data to a server or pull data from the server. For example, a PDA might pull a file from a laptop, or a phone synchronizing an address book might push it to a PDA. The similarities between the two communications protocols' lower layers mean that IrDA's OBEX protocol is ideally suited to transferring objects between Bluetooth devices.

***The Telephony Control Protocol* -** Bluetooth's Telephony Control protocol Specification (TCS) defines how telephone calls should be sent across a Bluetooth link. It gives guidelines for the signaling needed to set up both point to point and point to multipoint calls. By use of TCS, calls from an external network can be directed to other Bluetooth devices. For instance, a cellular phone could receive a call and use TCS to redirect the call to a laptop, allowing the laptop to be used as a hands-free phone. TCS is driven by a telephony application which provides the user interface, and provides the source of voice or data transferred across the connection set up by TCS.

***Audio / Video Control Transport Protocol (AVCTP*):** AVCTP describes the transport mechanisms to exchange messages for controlling A/V devices. This protocol adopts the AV/C device model and command format for control messages and those messages are transported by the Audio/Video Control Transport Protocol (AVCTP).

***Audio / Video Distribution Transport Protocol (AVDTP*):** AVDTP defines A/V stream negotiation, and transmission procedures.

5.2 BLUETOOTH PROFILES

In addition to protocols which guarantee that two units speak the same language, Bluetooth specification defines the profiles. They are associated with applications. The profiles specify which protocol elements are mandatory in certain applications. This concept prevents devices with little memory and processing power implementing the entire Bluetooth stack when they only require a small fraction of it. Simple devices like a headset or mouse can thus be implemented with a strongly reduced protocol stack.

The Bluetooth profiles are organized into groups, with each profile building upon the one beneath and inheriting features from below. For developers, this means that key features of one Bluetooth solution can be reused in other solutions, bringing down development costs and speeding up the development cycle.

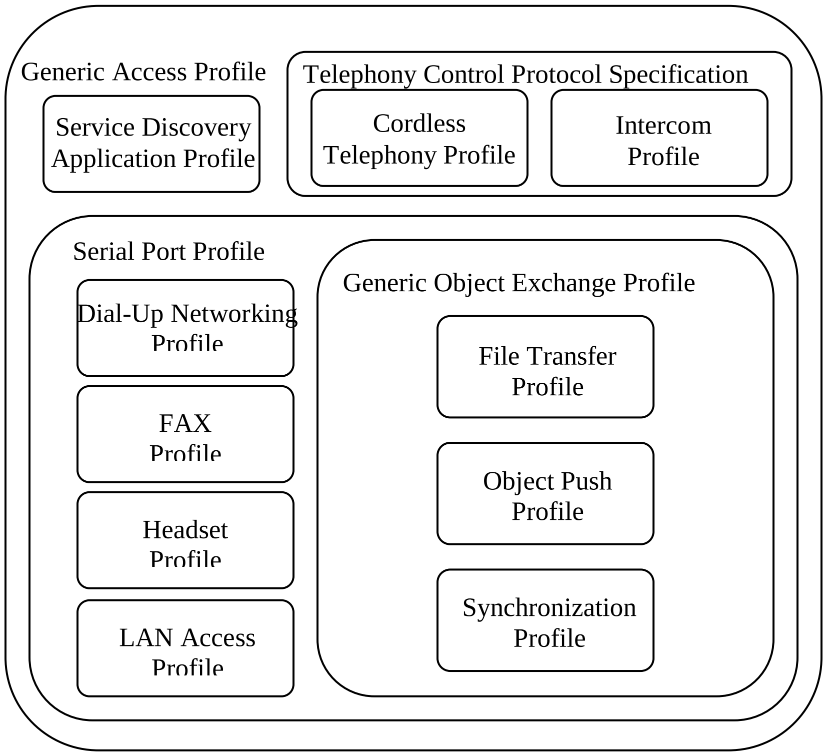


Figure 8: Bluetooth profiles

[Ref: GAP Profile Specfcation-www.bluetooth.org]

5.2.2 Basic Bluetooth Profiles:

Generic Access - It defines the basic rules for using the protocol stack. Provides the basis for all other profiles. GAP defines how two Bluetooth units discover and establish a connection with each other.

Serial Port - How to use RFCOMM's serial port emulation capabilities in Bluetooth products.

Dial-up Networking - A Bluetooth link to a modem.

Headset - A duplex link to a headset, controlled by an audio gateway such as cellular phone.

Generic Object Exchange - A set of rules for using OBEX, which supports file transfer, object push and synchronization profiles.

File Transfer - Transferring files between Bluetooth devices.

Object Push - Pushing objects from a Bluetooth enabled server to a client.

Synchronization - Synchronizing objects between Bluetooth devices.

Advanced Audio Distribution Profile (A2DP) : A2DP describes how stereo quality audio can be streamed from a media source to a sink. The profile defines two roles of an audio source and sink

Audio/Video Remote Control Profile (AVRCP): AVRCP is designed to provide a standard interface to control TVs, hi-fi equipment, or other to allow a single remote control (or other device) to control all the A/V equipment that a user has access to.

HFP: This is commonly used to allow car hands-free kits to communicate with mobile phones in the car. It commonly uses Synchronous Connection Oriented link (**SCO**) to carry a monaural audio channel with continuously variable slope delta modulation or pulse-code modulation.

Message Access Profile (MAP): specification allows exchange of messages between devices. Mostly used for automotive hands free use. The MAP profile can also be used for other uses that require the exchange of messages between two devices.

Bluetooth Low Energy Profiles:

GATT: Provides profile discovery and description services for Bluetooth Low Energy protocol. It defines how ATT attributes are grouped together into sets to form services.

CHAPTER-6

BLUETOOTH SECURITY

Trusted Device: The device has been previously authenticated, a link key is stored and the device is marked as "trusted" in the Device Database.
Untrusted Device: The device has been previously authenticated, a link key is stored but the device is not marked as "trusted" in the Device Database
Unknown Device: No security information is available for this device, e.g. **untrusted**

6.1 SECURITY MODES:

Mode 1: **No Security**
Only security at this level is by the nature of the connection: data-hopping and short-distance.
Bluetooth devices transmit over the unlicensed 2.45GHz radio band, the same band used by microwave ovens and cordless phones.
All Bluetooth devices employ “data-hopping”, which entails skipping around the radio band up to 1600 times per second, at 1MHz intervals (79 different frequencies).
Most connections are less than 10 meters, so there is a limit as to eavesdropping possibilities.

Mode 2: **Service Level Security**
Trusted devices have unrestricted access to all services, fixed relationship to other devices

1. Untrusted devices generally have no permanent relationship and services that it has access to are limited. Unfortunately, all services on a device are given the same security policy, other than application layer add-ons.

Services can have one of 3 security levels:
Level 3: Requires Authentication and Authorization. PIN number must be entered.
Level 2: Authentication only, fixed PIN ok.
Level 1: Open to all devices, the default level. Security for legacy applications, for example.

Mode3: **Link Level Security**

Security is implemented by symmetric keys in a challenge-response system.
Security implementations in Bluetooth units are all the same, and are publicly available:
Critical ingredients: PIN, BD_ADDR, RAND(), Link and Encryption Keys.

Security Entity:

PIN: up to 128 bit number, can be fixed (entered in only one device), or can be entered in both devices. If fixed, much lower security.
BD_ADDR: Bluetooth device address, unique 48 bit sequence. (IEEE). Devices must know the address of devices it wants to communicate with. Addresses are publicly available via Bluetooth inquiries.
Private Authentication Keys, or **Link Keys**: 128-bit random numbers used for authentication purposes. Paired devices share a link key.
Private Encryption Key: varying length key (8-128 bits), regenerated for each transmission from link key
RAND: frequently changing 128-bit random number generated by the device (in software). Common input for key generation. All Bluetooth devices have this random number generator.

6.2 SECURITY INITIALIZATION

1) Generation of initialization key
2) Authentication
3) Generation of link key
4) Link key exchange
5) Generation of encryption key in both devices.

Generation of initialization key

F(PIN, sizeof(PIN), RAND, BD_ADDR) produces 128 bit initialization key via shifting and xors (Linear feedback shift registers, whose output is combined by a state machine)
Device A and B now share the initialization key, which they use as their temporary link key while deciding on what kind of link key they will use for data transmission.
This key is discarded once they agree on a link key.

Authentication:

Does not always need to be mutual, specified by app, If it is mutual, then both act as verifiers, one after the other.
Device A: verifier
Device B: claimant
Basically determines if both have same shared key (ACO generated at this time as well for encryption. A issues challenge c to B, generated by its RAND A and B both run the RAND thru same function:
E1(c, BD_ADDR of B, current link key)
B sends its response to A, who checks to see that they match. If failure, start exponential waiting with a limit set on number of possible attempts.
On success, the BD_ADDR of other device is stored in the Device Database by the Service Manager.

Link Key:

LINK KEY GENERATION:
Unit key does not change; it was made when device was installed.Application decides which device will provide its unit key as a link key (favors the device with less memory). Shared initialization key is used to protect the transaction: it is XORed with the new link key.

LINK KEY EXCHNAGE:
After the unit key is stored on the other device, the initialization key is discarded.Higher security: combination key is used rather than the unit key, and this is formed by F(unit key, RAND, BD_ADDR) on both A and B.Master-slave communications use Master link key. A slave gets a master link key when first connected to Master and then changes it when prompted by Master.

Encryption:

Encryption requires an authenticated link with an established link key
Devices must agree on an encryption key to communicate. Packet payloads are encrypted (not the packet headers or access codes). Devices negotiate on what size Encryption key they need, typically around 64 bits. Range is 1-16 bytes.

Encryption Modes**:**
Encryption Mode depends on the shared key:

If unit or combination key, then point-to-multipoint traffic is not encrypted. Individual traffic may or may not be encrypted (app specific) If a master key is used, there are three possible modes:
Mode 1: nothing is encrypted.
Mode 2: broadcast traffic is not encrypted, but the individually addressed traffic is encrypted with the master key.
Mode 3: all traffic is encrypted with the master key.

Encryption Process.
Key = E3(COF, RAND, LinkKey). Variable size design due to internationalization issues
COF: Ciphering Offset Number, determined by type of link key:
Combination/Unit Link Key: equal to ACU from initialization This was obtained during the initialization key generation process and saved in the Security Manager.
Master Link Key: Concatenation of the master BD_ADDR and the slave BD_ADDR

6.3 BLUETOOTH KEYS:

Encryption Key is not a Link Key but is derived from either the Unit, Combination, or Master Key. Can be shorter than the 128-bit link keys.

Four Link Keys:

Ki : initialization key, protects initialization parameters. Formed from combo of RAND, PIN, and BD_ADDR. This is discarded after channel establishment, at which point the others are used.

Kab: combination key, derived from paired devices when additional security needed. Memory? Device that has the most has to store the combo key

Ku: unit key, generated in installation of a device, stored in nonvolatile memory. Unit and combo keys generated with the same function, different inputs.

Unit Key cannot change! If it does, then the entire piconet is compromised and must be re-initialized

Km: master key, used when the master device wants to transmit to multiple devices at once. Overrides current link keys for one session.
Master Key is the most typical link key, but for scatternet communication (multiple masters), the unit key or combination key is always used.

CHAPTER-7

Data Transfer Technique

As already explained in section above there are multiple Bluetooth profile's designed for specific target and to achieve a specific functionality. Each profile has its own set of rules and specification that has to be followed by all profile implementers to achieve that functionality. This ensures interoperability between different operators. Based on BT SIG supported profiles, below set of profiles can be used for data transfer. Each profile has specific data types or format which it can transfer; some operates on bits, while other on files and PIMS object etc.
Protocols followed by each of below profiles are different. Some profiles are designed to operate on large size of data while others can transfer fix size of bytes only.
Each profile's has its own set of authentication or encryption technique. Each of these profiles has been discussed in details in below section.

- OBJECT PUSH PROFILE
- FILE TRANSFER PROFILE
- SERIAL PORT PROFILE
- BT LOW ENRGY (BT 4.0)

CHAPTER-8

OBJECT PUSH PROFILE

8.1 Introduction

The Object Push Profile (OPP) defines the requirements for the protocols and procedures that shall be used by the applications providing the Object Push usage model. This profile makes use of the Generic Object Exchange Profile (GOEP) [10], to define the interoperability requirements for the protocols needed by applications. Common devices using these models are notebook PCs, PDAs, and mobile phones.

The scenarios covered by this profile are the following:

• Use of a Bluetooth device to push an object to the inbox of another Bluetooth device. The object can for example be a business card or an appointment.

• Use of a Bluetooth device to pull a business card from another Bluetooth device.

• Use of a Bluetooth device to exchange business cards with another Bluetooth device. Exchange is defined as a push of a business card followed by a pull of a business card.

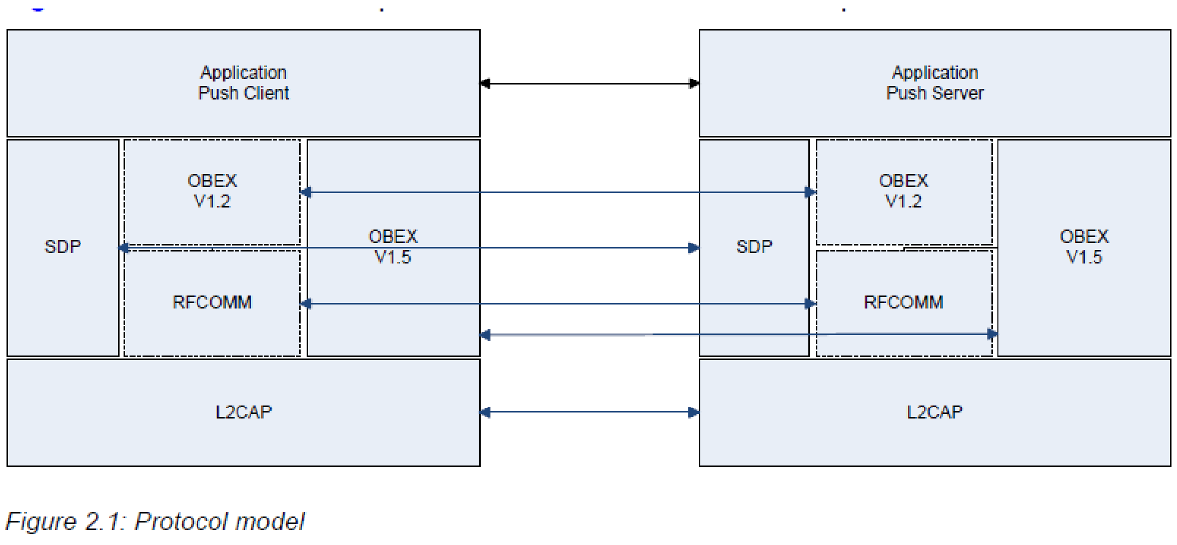


FIGURE 9: OBEX Protocol Model
[Ref: OPP Profile Specification-www.bluetooth.org]

8.2 Configurations and Roles

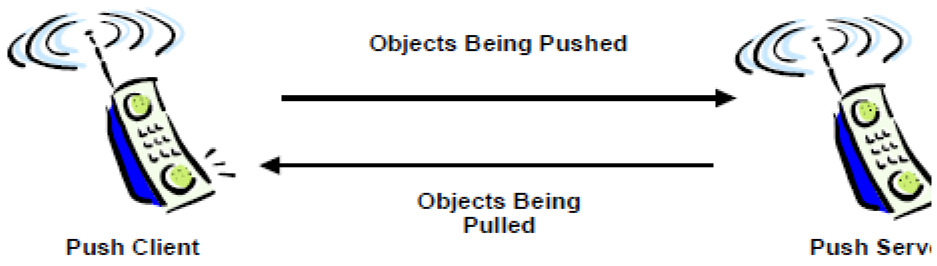


Figure 2.2: Push and Pull Example between Two Mobile Phones

FIGURE-10: OPP PUSH and PULL OPERATIONS
[Ref: OPP Profile Specification-www.bluetooth.org]

The following roles are defined for this profile:

Push Server – This device provides an object exchange server. In addition to the interoperability requirements defined in this profile, the Push Server shall comply with the interoperability requirements for the server of the GOEP if not defined in the contrary.

Push Client – This device pushes and pulls objects to and from the Push Server. In addition to the interoperability requirements defined in this profile, the Push client shall also comply with the interoperability requirements for the client of the GOEP, if not defined to the contrary.

8.3 Profile Scenarios:

The scenarios covered by this profile are:

• Use of a Push Client to push an object to a Push Server. The object can, for example, be a business card or an appointment.
• Use of a Push Client to pull a business card from a Push Server.
• Use of a Push Client to exchange business cards with a Push Server.

The push operation described in this profile pushes objects from the Push Client to the inbox of the Push Server.

8.4 Profile Essential:

- Link level authentication and encryption are mandatory to support and optional to use. When using Core Specification v2.1 or later, encryption is mandatory to use; otherwise its use is optional.
- Bonding is mandatory to support and optional to use.
- OBEX authentication shall not be used.
- This profile does not mandate the server or client to enter any discoverable or connectable modes automatically, even if they are able to do so.
- On the Push Client side, end-user interaction is always needed to initiate an object push, business card pull or business card exchange.

8.5 Profile Function:

There are three different **functions** associated with the Object Push profile:

- Object Push function
- Business Card Pull function
- Business Card Exchange function

The **Object Push function** initiates the function that pushes one or more objects to a Push Server.

The **Business Card Pull function** initiates the function that pulls a business card from a Push Server.

The **Business Card Exchange function** initiates the function that exchanges business cards with a Push Server.

The three functions should be activated by the user.
When the user selects one of these functions, an inquiry procedure may be performed to produce a list of available devices in the vicinity

CHAPTER-9

File Transfer Profile

9.1 Introduction

This profile defines the requirements for the protocols and procedures that shall be used by the applications providing the file transfer usage model. The file transfer usage model makes use of the underlying Generic Object Exchange Profile (GOEP) to define the interoperability requirements for the protocols needed by applications. Typical scenarios covered by this profile involving a Bluetooth device browsing , transferring and manipulating objects on/with another Bluetooth device.

9.2 Profile Overview

9.2.1 Roles/Configurations

The following roles are defined for this profile:
Server – The Server device is the target remote Bluetooth device that provides an object exchange server and folder browsing capability using the OBEX Folder Listing format. In addition to the interoperability requirements defined in this profile, the Server must comply with the interoperability requirements for the Server of the GOEP if not defined in the contrary.
Client – The Client device initiates the operation, which pushes and pulls objects to and from the Server. In addition to the interoperability requirements defined in this profile, the Client must also comply with the interoperability requirements for the Client of the GOEP if not defined in the contrary.

9.2.2 Profile Scenarios

The scenarios covered by this profile are the following:
Usage of the Client to browse the object store of the Server. Clients are required to pull and understand Folder Listing Objects. Servers are required to respond to requests for Folder Listing Objects. Servers are required to have a root folder. Servers are not required to have a folder hierarchy below the root folder.

Usage of the Client to transfer objects to and from the Server. The transfer of objects includes folders and files. Clients must support the ability to push or pull files from the Server. Clients are not required to push or pull folders. Servers are required to support file push, pull, or both. Servers are allowed to have read-only folders and files, which means they can restrict object pushes. Thus, Servers are not required to support folder push or pull.

Usage of the Client to create folders and delete objects *(folders and files)* on the Server. Clients are not required to support folder/file deletion or folder creation. Servers are allowed to support read-only folders and files, which means they can restrict folder/file deletion and creation. A device adhering to this profile must support Client capability, Server capability or both.

9.2.3 Profile Fundamentals

The profile fundamentals are the same as the GOEP
Support for link level authentication and encryption is required but their use is optional. Support for OBEX authentication is required but its use is optional. Support of bonding is required but its use is optional.
This profile does not mandate the server or client to enter any discoverable or connectable modes automatically, even if they are able to do so. On the Client side, end-user intervention is always needed to initiate file transfer.

9.3 User Interface Aspects

File Transfer Mode Selection, Servers

Servers must be placed in File Transfer mode. This mode enables a Client to perform file transfer operations with the Server. When entering this mode, File Transfer Servers should set the device in Limited Discoverable mode (see Generic Access Profile), ensure that the Object Transfer Bit is set in the CoD, and register a service record in the SDDB. It is recommended that this mode be set and unset by user interaction, when possible.

Function Selection, Clients

Clients provide file transfer functions to the user via a user interface. An example of a file transfer user interface is a file-tree viewer to browse folders and files. Using such a system file-tree viewer, the user can browse and manipulate files on another PC, which appears in the network view.

File Transfer Applications provide the following functions.

- Select Server
- Navigate Folders
- Pull Object
- Push Object

- Delete Object
- Create Folder

When the user selects the Select Server function, an inquiry procedure will be performed to produce a list of available devices in the vicinity.

9.4 Profile Function

This section describes the service capabilities which can be utilised by the application profiles using GOEP.

Feature Overview

The Folder Browsing function and Object Transfer (File Transfer) is mandatory on both the File Transfer Client and File Transfer Server. The Object Transfer (Folder Transfer) and Object Manipulation Function are optional. However the server must be able to respond with an error code on any request, even if it doesn't support this feature

Folder Browsing

A file transfer session begins with the Client connecting to the Server and pulling the contents of the Server's root folder. When an OBEX connection is made, the Server starts out with its current folder set to the root folder.

Browsing an object store involves displaying folder contents and setting the 'current folder'. The OBEX SETPATH command is used to set the current folder. To display a folder hierarchy starting with the root folder, the Client must read the root folder contents using GET. It must then retrieve the contents of all sub-folders using GET. If the sub-folders contain folders, then the Client must retrieve the contents of these folders and so on. To retrieve the contents of a folder, the Client must set the current folder to the sub-folder using SETPATH, then pull the sub-folder contents using GET.

Object Transfer

Objects are transferred from the Client to the Server using OBEX PUT, and objects are transferred from the Server to the Client using OBEX GET. Transferring files requires a single PUT or GET operation per file. Transferring folders requires transferring all the items stored in a folder, including other folders. The process of transferring a folder may require that new folders be created. The SETPATH command is used to create folders.

Object Manipulation

A Client can delete folders and files on a Server. It can also create new folders on a Server. A brief summary of these functions is shown below.

A file is deleted by using a PUT command with the name of the file in a Name header and no Body header.

An empty folder is deleted by using a PUT command with the name of the folder in a Name header and no Body header.

A non-empty folder can be deleted in the same way as an empty folder but Servers may not allow this operation. If a Server refuses to delete a non-empty folder it must return the "Precondition Failed" (0xCC) response code. This response code tells the Client that it must first delete all the elements of the folder individually before deleting the folder.

A new folder is created in the Server’s current folder by using the SETPATH command with the name of the folder in a Name header. If a folder with that name already exists, then a new folder is not created. In both cases the current folder is set to the new folder.

CHAPTER-10

Bluetooth Low Energy

Bluetooth Low Energy (BLE) is designed to provide significantly lower power consumption. This allows Android apps to communicate with BLE devices that have low power requirements, such as proximity sensors, heart rate monitors, fitness devices, and so on
BTLE devices will go into sleep mode and wake only for connection attempts or events. As a result, the software developer needs to understand a few of the basic concepts in BTLE

10.1 Generic Attribute Profile (GATT)—The GATT profile is a general specification for sending and receiving short pieces of data known as "attributes" over a BLE link. All current Low Energy application profiles are based on GATT.

10.2. Attribute Protocol (ATT)—GATT is built on top of the Attribute Protocol (ATT). This is also referred to as GATT/ATT. ATT is optimized to run on BLE devices. To this end, it uses as few bytes as possible. Each attribute is uniquely identified by a Universally Unique Identifier (UUID), which is a standardized 128-bit format for a string ID used to uniquely identify information. The *attributes* transported by ATT are formatted as *characteristics* and *services*.

Characteristic—A characteristic contains a single value and 0-n descriptors that describe the characteristic's value. A characteristic can be thought of as a type, analogous to a class.

Descriptor—Descriptors are defined attributes that describe a characteristic value. For example, a descriptor might specify a human-readable description, an acceptable range for a characteristic's value, or a unit of measure that is specific to a characteristic's value.

Service—A service is a collection of characteristics. For example, you could have a service called "Heart Rate Monitor" that includes characteristics such as "heart rate measurement

10.3 GATT Operations

These are all commands a client can use to discover information about the server.

- Discover UUIDs for all primary services. This operation can be used to determine if device supports Device Information Service, for example.

- Discover all characteristics for a given service. For example, some heart rate monitors also include a body sensor location characteristic.

- Read and write descriptors for a particular characteristic. One of the most common descriptors used is the Client Characteristic Configuration Descriptor. This allows the client to set the notifications to *indicate* or *notify* for a particular characteristic. If the client sets the Notification Enabled bit, the server sends a value to the client whenever the information becomes available. Similarly, setting the Indications Enabled bit will also enable the server to send notifications when data is available, but the indicate mode also requires a response from the client.

- Read and write to a characteristic. Using the example of the heart rate monitor, the client would be reading the heart rate measurement characteristic. Or, a client might write to a characteristic while upgrading the firmware of the remote device.

10.4 BLE USE CASES:

Blood pressure profile. Proximity Profile, Heart Rate Profile etc. Each profile is designed with fix set of characteristics having unique UUID.

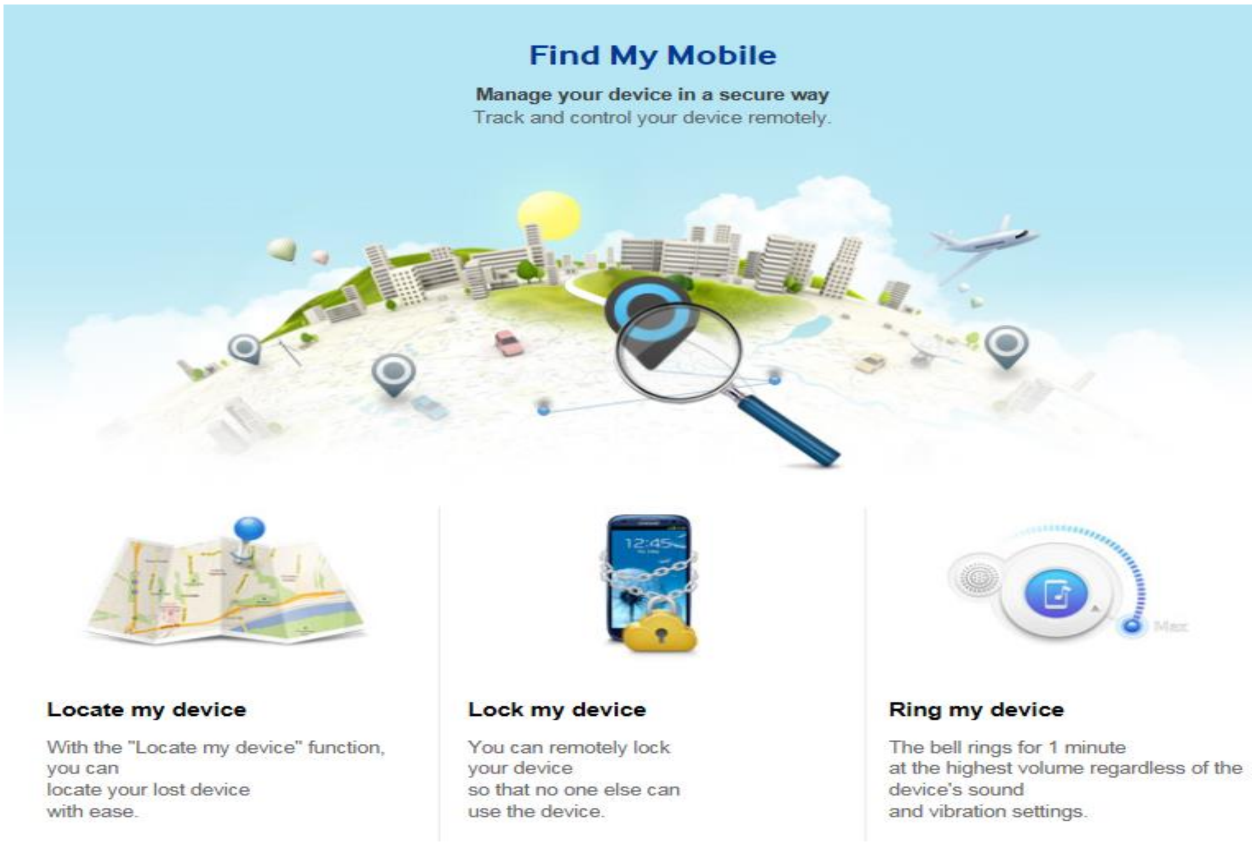
10.5 BLE LIMITATION:

As BLE is designed to keep connection active for very small time . so only small brust of data Can be transferred using BLE in form of characteristics and descriptors as ATT protocol designed.

CHAPTER: 11

Short Range Wireless Tracker

Mobile is getting very popular and becoming utility device. It has been used various purpose as Multimedia device, play station, radio and other processing unit. In case device gets lost then user wants to get intimation of its utility device to trace or get it last presence and usage.



Find My Mobile

Manage your device in a secure way

Track and control your device remotely.

Locate my device

With the "Locate my device" function, you can locate your lost device with ease.

Lock my device

You can remotely lock your device so that no one else can use the device.

Ring my device

The bell rings for 1 minute at the highest volume regardless of the device's sound and vibration settings.

FIGURE11: Mobile Tracker

Most of method usage can work if it has on network or data network to check it last usage tracking system. Now question come in case lost device does not comes to network and sim network so no way to get any intimation of that device. Since smart device has many usage as utilities service like multimedia, gaming and other usage.

Here is a new approach to tracking based on short range wireless technology. All tracking capable device will be running a client which will be seeking for an advertise message for loss indication on regular basis. Once a device has been used for without SIM or Data connectivity (over WIFI or other means), will be treated as a lost possible category and will be prompt to enter a user define credential if not provided correctly will be treated as a lost device.

11.1 Lost Device Detection

This an important part of tracker system to detect device has been lost if this step is not clear user of device unnecessary

Warring message will be displayed and user will be prompt to enter his credential to device. It wills dependent on type of device and it availability of detection of lost.

Let take an example of mobile phone which has Sim card and other data network usage (like WIFI etc.) .So device primary usage will be SIM card insertion or on data network connection. One method of detection will be both Sim and data connectivity not has been present for more than 48 hours(just for example) then warning comes to enter credential if not enter properly for 2 time then device has been taken in lost category.

11.2 *Lost device Server and client*

In this model there are two entities come in center roll of tracking.one is lost device server and another one is normal device client. All the device how will be tracking enable will consist of both the entities in device and one will active both the device status like lost category or normal category. In normal category device normal client will be running and lost device server will be sleeping state, on other hand in lost device category lost device server will be running and normal device client will be in sleeping state.

Normal device client: This service or entity will take care for any help message detection of broadcasting message in the short range vicinity. Once it get help message from vicinity and connect to lost device server to get available unique register information ,sending confirmation message to lost device for help processing ack.

Lost device Server: This service or entity will take care for broadcasting help message in the short range vicinity. Once any normal device client wants connect to allow that connection, sharing unique register information and get confirmation from Peer

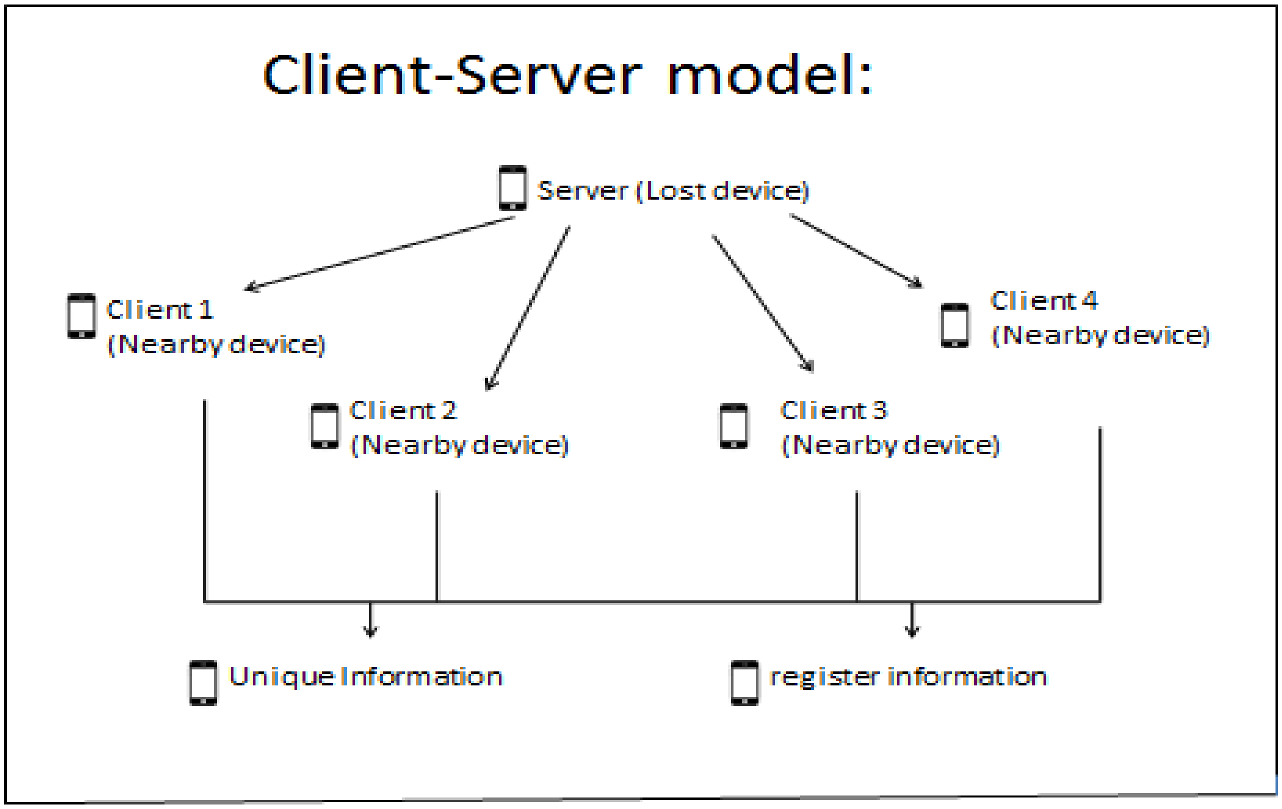


FIGURE12: Client –Server Model

11.3 Working architecture

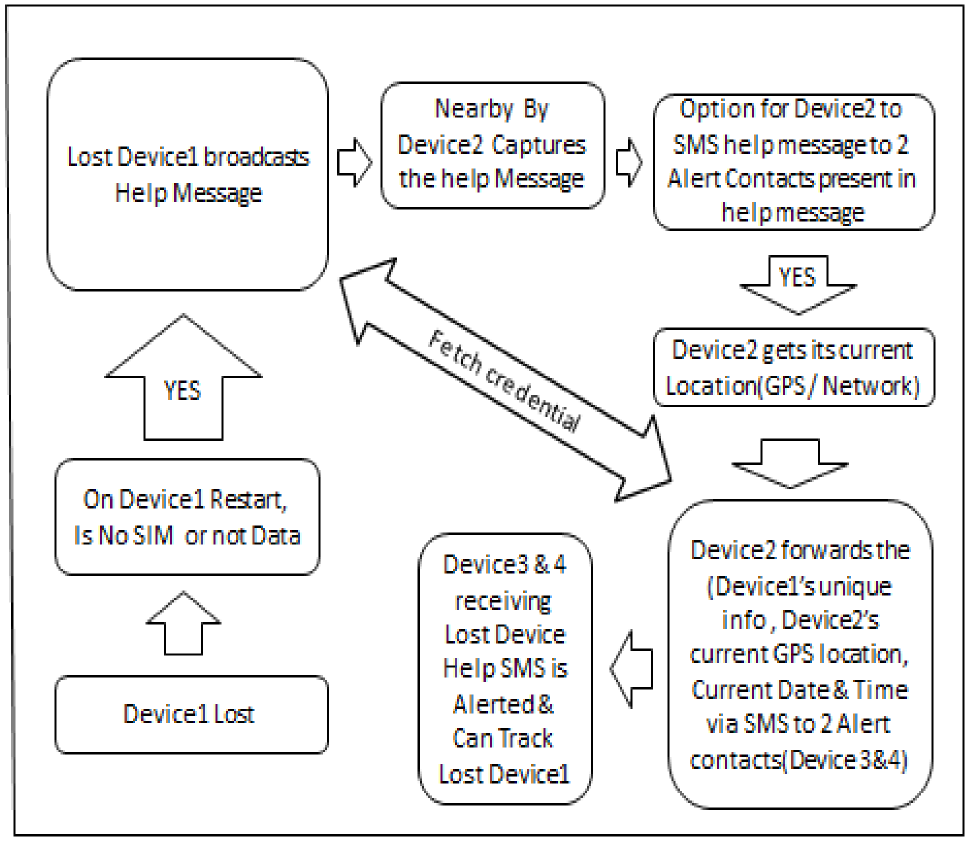

Lost Device1 broadcasts Help Message
Nearby By Device2 Captures the help Message
Option for Device2 to SMS help message to 2 Alert Contacts present in help message
YES
Device2 gets its current Location(GPS / Network)
Fetch credential
Device2 forwards the (Device1's unique info , Device2's current GPS location, Current Date & Time via SMS to 2 Alert contacts(Device 3&4)
Device3 & 4 receiving Lost Device Help SMS is Alerted & Can Track Lost Device1
YES
On Device1 Restart, Is No SIM or not Data
Device1 Lost

FIGURE13: Basic working Model

All tracking capable device will be running a client which will be seeking for an advertise message for loss indication on regular basis. Once a device has been used for without SIM or Data connectivity (over WIFI or other means), will be treated as a lost possible category and will be prompt to enter a user define credential if not provided correctly will be treated as a lost device.

CHAPTER: 12

APPLICATION ARCHITECTURE

This model can be implemented using short range technology blue tooth. Since blue tooth is basically available in mobile of smart and utilities device and operates on 2.4 GHz which is unlicensed free to use. Blue tooth have feature to support client server model and has UUID support for specific profiles and service. Also blue tooth operations are low power consumption which can in turn full fill running client and server in devices.

Now there are two type of implementation possible using traditional blue tooth BT EDR and Blue tooth Low energy 4.0.

12.1 TRADITIONAL BLUETOOH BT EDR

In this method new lost profile need to be implemented for tracker. New lost profile server will start service /profile discovers to know search device supported profile. Once it get device support lost profile then it can send help message to supported client. Client accept

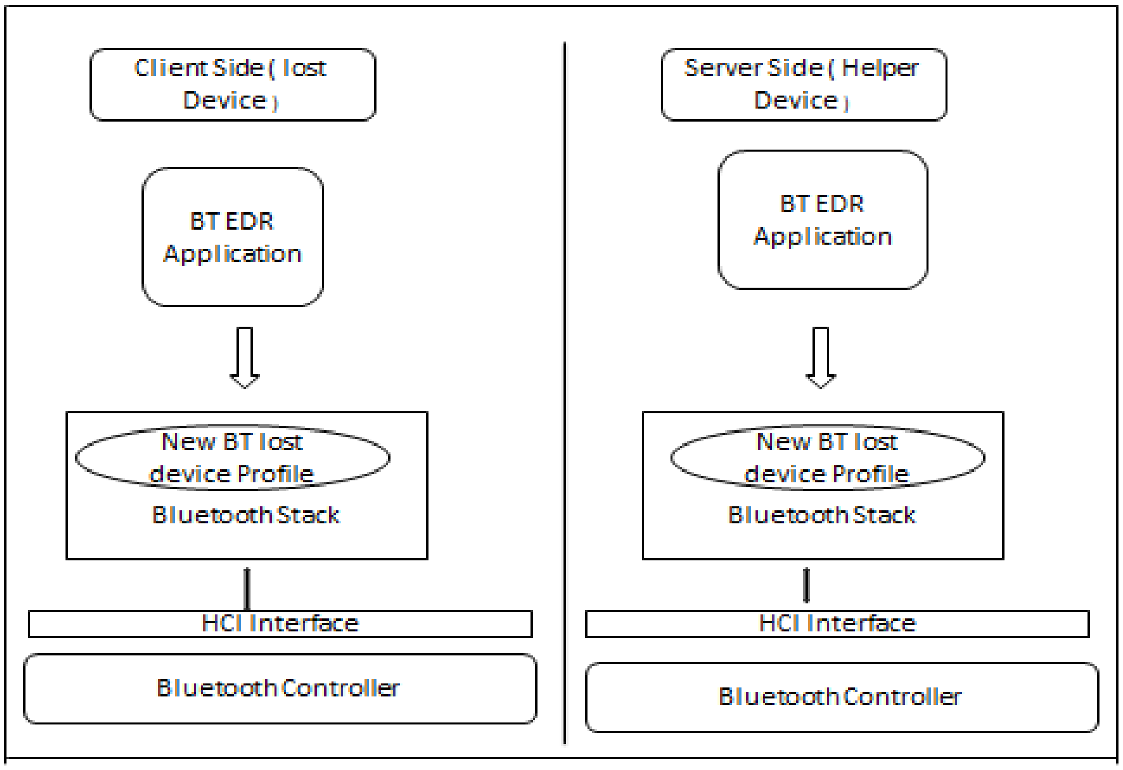


FIGURE14: BT EDR architecture

help packet and start connection with server and server send unique information to client. Then client send ACK to final information and close connection.

12.1.1 Bluetooth Lost profile:

Blue tooth Lost profile: This is new proposed profile to handle lost device handling in Blue tooth EDR case. This profile is need for detection of device which is capable of support lost detection. This profile will consider 2 role, Server Role and client role. Both the role will have different Service UUID.

Server Role: This role should be active in all the devices which need to help lost device. Server role would support server discovery and connection with client.

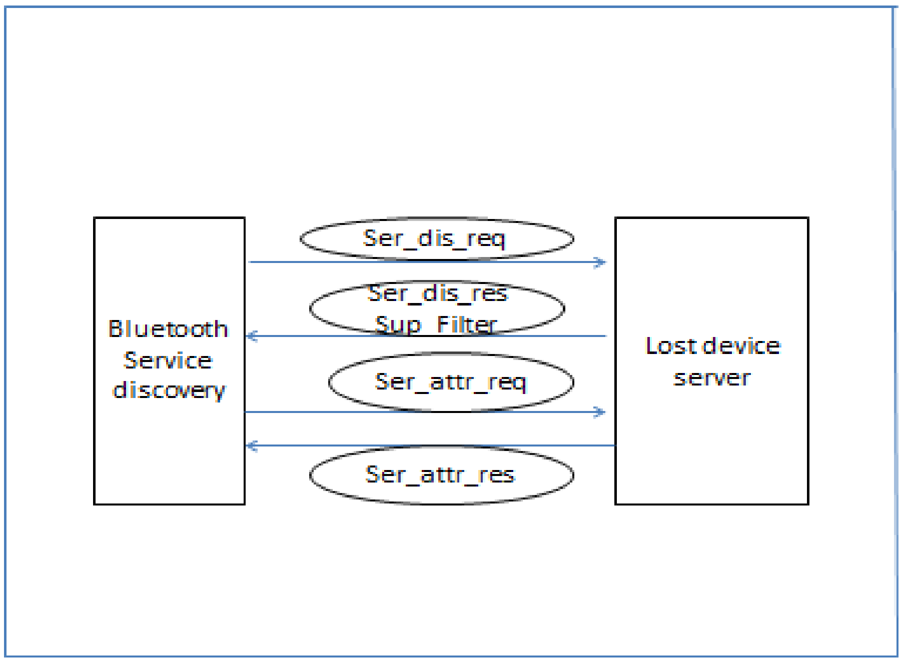


FIGURE15: Lost Profile service discovery

Client Role: This role should be active in all the devices which have been detected lost device. Client role would support server discovery and connection with client.
Client role will start device discovery followed by service discovery. Server role service discovery has been shown in above fig.

Once client get the device and service discovery it follow below step.
Step 1: client start transport level connection followed by service level connection.

Step 2: Based on success res for connection request .client start obex session with server.
Step 3: Based on Obex session creation client start put to server for unique data .

Based on the above step a lost device client can start connection with server and exchange unique information based on connection conformation.
The unique data exchange can use encryption to hacking over the air.

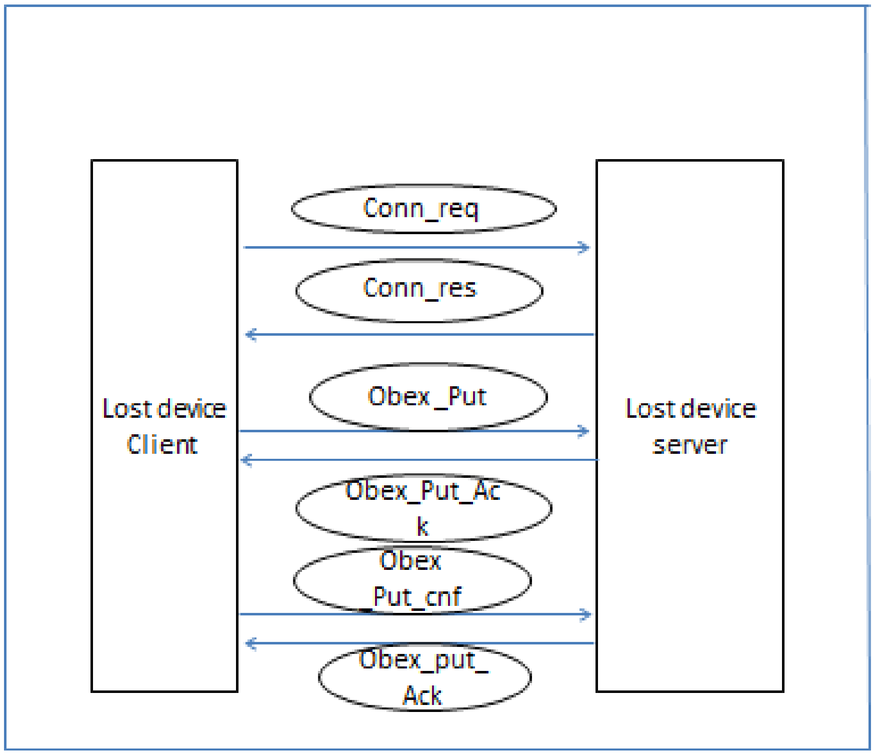


FIGURE16: Lost Profile Connection

BT EDR Application: BT Application manager both the role and takes decision for activating roles. This also interacts with other Module. Here is role of BT EDR Application role .

1. Role selection: Application check weather device has been lost or not initiate roles like Server or client.
2. Searching interval : Application also decide scan interval for server search in case client role running in background
3. Application communicate to other Module like GPS etc to get information and embedded with unique information and send to register number or location
4. Application fetch unique information encodes and decode on sending and receiving respectively.

12.2 BLUETOOTH LOW ENERGY

In this implementation no need of new profile implementation as Blue-tooth low energy support custom UUID which can be used for tracking implementation. A custom UUID can be decided to use for this supposed Lost server application APK will start broadcasting help message on custom UUID X. Normal device client will received this help advertise and check UUID in it is on X UUID then it will recognize lost device. So client will fetch unique info and register information from the advertise message. Client will make SMS with unique information and GPS location of client device and send to register number.

12.2 .1 Peripheral /Central role:

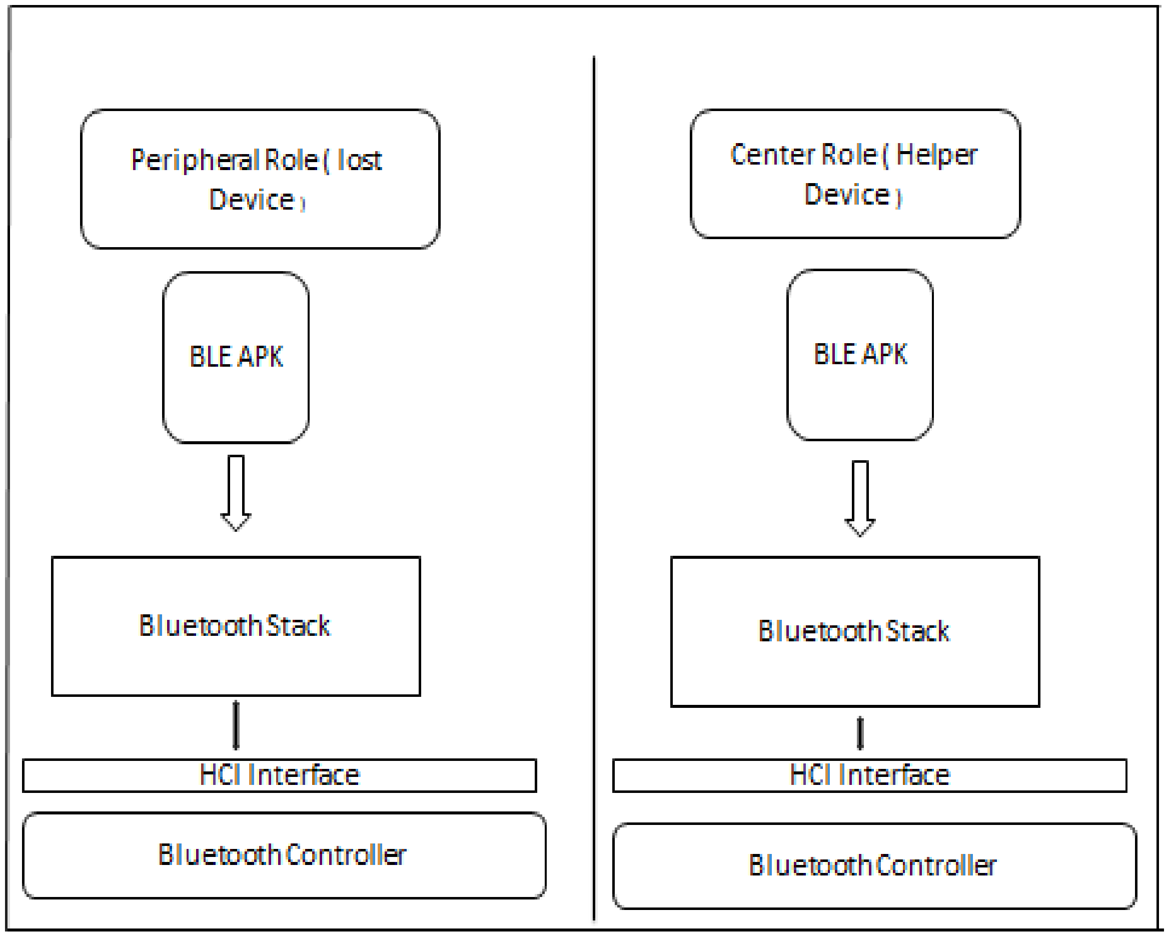
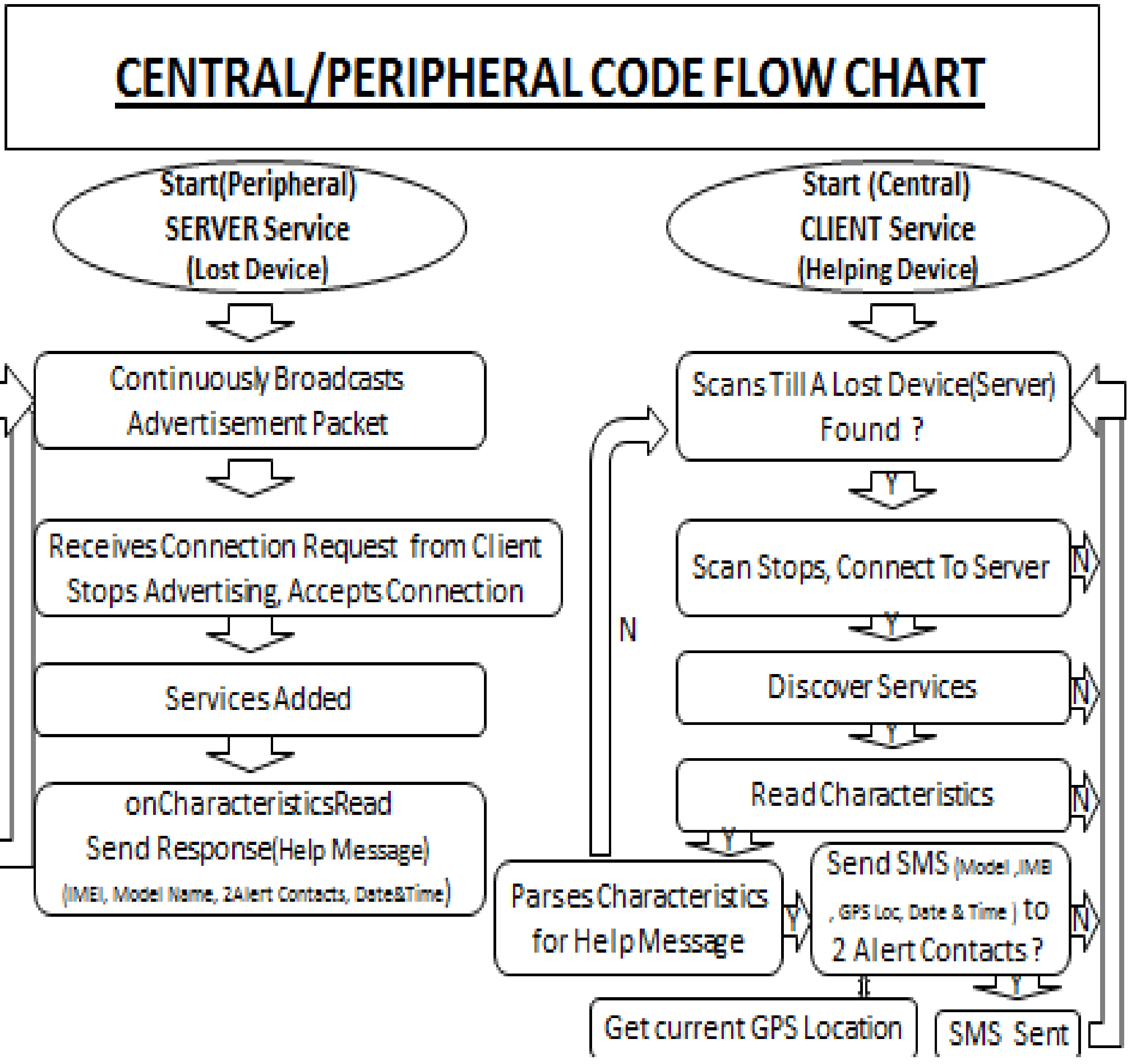


FIGURE17: Peripheral/Central Role architecture

Code Flow for Peripheral /Central role:



CENTRAL/PERIPHERAL CODE FLOW CHART
Start(Peripheral) SERVER Service (Lost Device)
Continuously Broadcasts Advertisement Packet
Receives Connection Request from Client Stops Advertising, Accepts Connection
Services Added
onCharacteristicsRead Send Response(Help Message) (IMEI, Model Name, 2Alert Contacts, Date&Time)
Start (Central) CLIENT Service (Helping Device)
Scans Till A Lost Device(Server) Found ?
Y
Scan Stops, Connect To Server
N
Y
Discover Services
N
Y
Read Characteristics
N
Y
N
Parses Characteristics for Help Message
Y
Send SMS (Model ,IMEI , GPS Loc, Date & Time) to 2 Alert Contacts ?
N
Y
Get current GPS Location
SMS Sent

FIGURE18: Peripheral/Central Role code Flow

12.2 .1 Broadcast / Observer role:

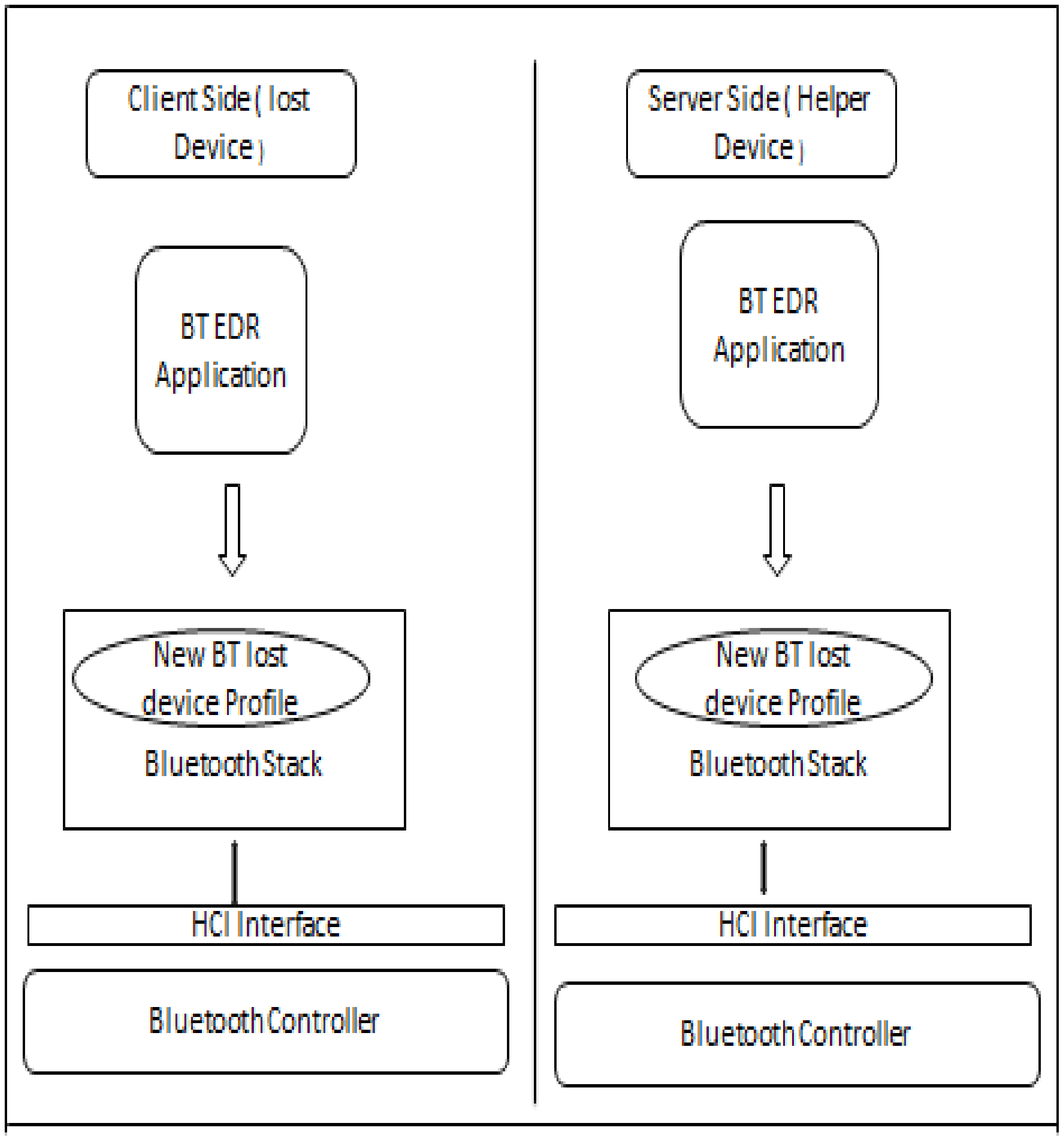


FIGURE19: Broadcast / Observer Role architecture

Code Flow for Broadcast /Receiver role:

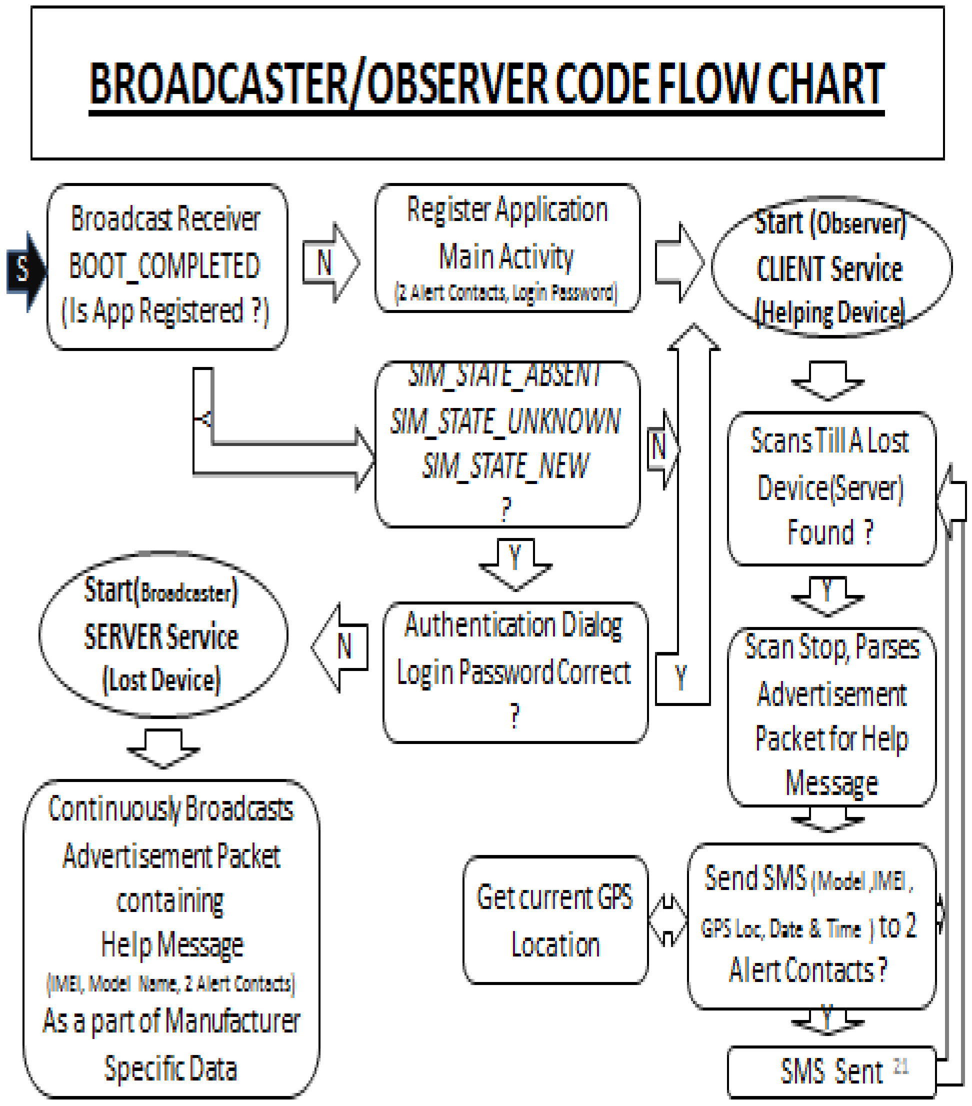
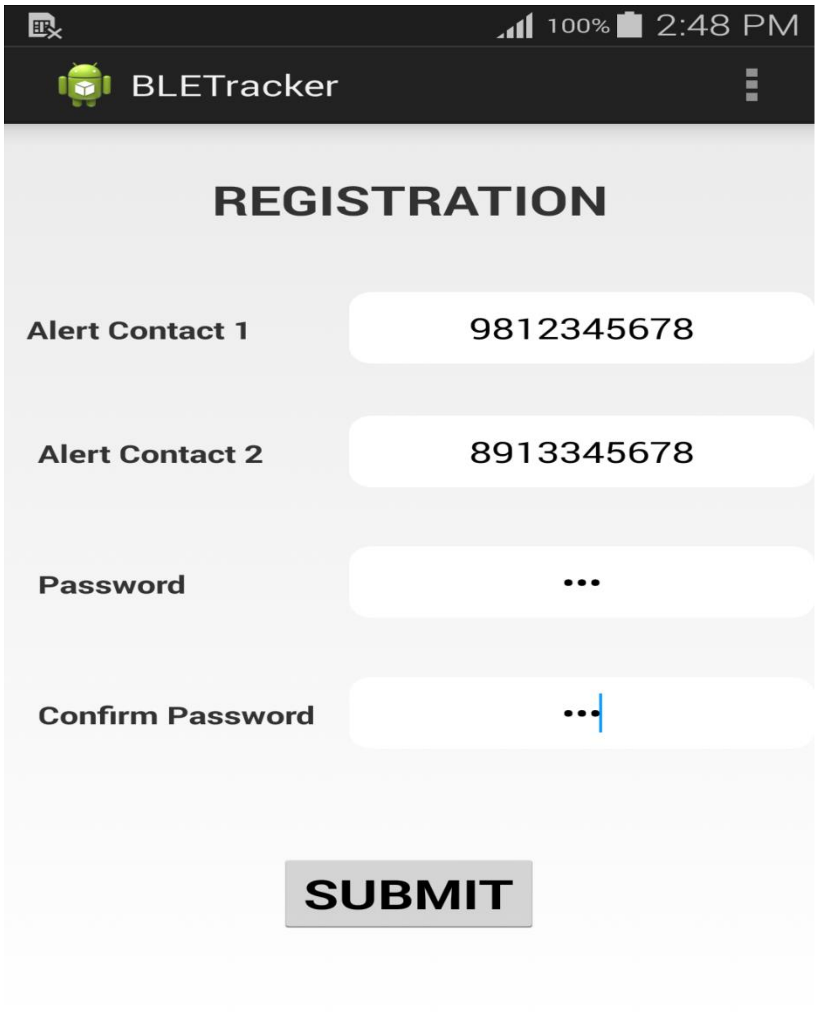

BROADCASTER/OBSERVER CODE FLOW CHART
S
Broadcast Receiver BOOT_COMPLETED (Is App Registered ?)
N
Register Application Main Activity (2 Alert Contacts, Login Password)
Start (Observer) CLIENT Service (Helping Device)
SIM_STATE_ABSENT SIM_STATE_UNKNOWN SIM_STATE_NEW ?
N
Y
Scans Till A Lost Device(Server) Found ?
Start(Broadcaster) SERVER Service (Lost Device)
N
Authentication Dialog Login Password Correct ?
Y
Y
Scan Stop, Parses Advertisement Packet for Help Message
Continuously Broadcasts Advertisement Packet containing Help Message (IMEI, Model Name, 2 Alert Contacts) As a part of Manufacturer Specific Data
Get current GPS Location
Send SMS (Model, IMEI, GPS Loc, Date & Time) to 2 Alert Contacts ?
Y
SMS Sent 21

FIGURE19: Broadcast / Observer Role Code Flow

CHAPTER: 13

BLUTOOT LOW ENERGY APP – MAIN PAGE



REGISTRATION

Alert Contact 1	9812345678
Alert Contact 2	8913345678
Password	•••
Confirm Password	•••

FIGURE21: APPLICATION MAIN PAGE

CHAPTER: 14

CONCLUSION

We have presented a wireless tracking system over Blue tooth short range technology using traditional and low energy. Both methods can be compare based on attributes like Performance, Modifiability, and Portability, Usability, Extensibility, and Power consumption .Here is compression study on both version of implementation.

Attribute	Traditional Bluetooth EDR	Bluetooth Low Energy
Performance	Good	Better as less overhead
Modifiability	Good	Better as easy to upgrade
Portability	Portability will be complex in terms of addition or deletion of functionality	Easy to Port no major dependency on Stack
Usability	Good	Good
Extensibility	Relative complex	Simple
Power consumption	High	Very less (1/1000)

CHAPTER: 15

FUTURE WORK

Device lost is major problem and keep on increasing due to difficult to track utility device. This application will be enhanced in adding below control in detecting low device on wireless method.

1. Adding control to client device to play ring tone on the lost mobile .This will really help to identify the potential thief as since both the client and server   in 10 meter range.

2. Adding control in App to get a silent mode Photo capture on the server mobile and sent to client on. This will help to identify the potential thief.

3. Making this wireless detection technique in Bluetooth Sig so that mode of Bluetooth device support and easy to control.

4. Make user registration data in mobile or device trust Zone so that user information can be removed and detection work perfectly well.

5. New security mechanism can be use while sending unique date in air.

REFERENCES

i. Diego Melpignano1 and Andrea Zanella2 “"**Analysis of File Transfer Protocol Over Radio Link"** *12-th Tyrrhenian International Workshop on Digital Communications,Portoferraio - Isle of Elba (Italy), Sept. 13-16 2000*

ii. Pradeep Vishwakarma1, Professor Brajesh patel “Security– **Bluetooth Security– Secure Data Transfer over Bluetooth** IJCSMS International Journal of Computer Science and Management Studies, Vol. 11, Issue 02, Aug 2011

iii. Yujin Noishiki Hidetoshi Yokota Akira Idou et al “**Design and Implementation of Ad-hoc Communication and Application on Mobile Phone Terminals”** KDDI R&D Laboratories, Inc. 2-1-15 Ohara, Fujimino-Shi, Saitama

iv. Nikita Mahajan *et al*, **Design of Chatting Application Based on Android Bluetooth** International Journal of Computer Science and Mobile Computing, Vol.3 Issue.3, March- 2014,

v. Juha T. Vainio et al **“Bluetooth Security”** Helsinki University of Technology 2000-05-25

vi. The Bluetooth Special Interest Group.Bluetooth Specification Core v4.0.(2009-02).Http://www.bluetooth.org.