Evaluation of IP over Bluetooth

Graduate Masters Thesis

Siv.ing. Degree
in
Information and Communication
Technology

By

Håkon Gunleifsen and Kjell Rune Øyrås

Waterford, May 2002

Abstract

The increasing use of Wireless technologies suggest that this kind of technology has a bright future. Today the most popular Wireless Local Area Networks are WLAN 802.11 and Bluetooth. They both have their strengths in different environments. Bluetooth was not initially an Internet technology, but over the years the focus on implementing Bluetooth nodes with Internet capability has increased. When we want to run IP over Bluetooth we need some kind of gateway to the Internet from Bluetooth, a network access point. This will be an important device in the thesis. The Bluetooth Special Interest Group has suggested two ways to implement IP over Bluetooth in different scenarios, but there is still a lot more work to do. This thesis will deal with some of these approaches. Hence, we will discuss problems like: the Bluetooth protocol stack, scalability, efficiency and mobility, which are the factors considered important to Bluetooth and IP.
The thesis will give an evaluation of these different issues and in the end we will give a short outline of how we suggest making a larger Bluetooth network which runs over IP.

The Stack – There has been several suggestions about how to implement IP over Bluetooth. We have taken a look at four different designs of the Bluetooth stack and evaluate all of them. There are mainly two designs that are relatively good and can both be used. We will give preference to these two methods and their strengths and weaknesses.

Scalable – The number of Bluetooth devices within one single Bluetooth-network is currently quite limited (piconets). But according to the Bluetooth specification we can interconnect several networks (scatternets). It has not yet specified how to realize IP over Bluetooth, but there is work going on. If we had a network like this, the network will become more complex and slow, and maybe that's the reason this is not specified. We will evaluate why this is so and suggest ways to make larger Bluetooth networks more efficient.

Efficient – At the moment Bluetooth networks are relatively slow. The Bluetooth network is not made for high bit rate transmissions, but still there is some need for a certain bandwidth. By making a good designed protocol stack and a good network structure, we will try to make Bluetooth as efficient as possible.

Mobility – As a part of every wireless technology, we always set some requirements for roaming and mobility. We want Bluetooth devices to roam and connect to the nearest Internet access point. Issues we will need to discuss concerning this topics are methods to realize a well designed network that covers the needs of Bluetooth roaming. Some methods we will discuss are Mobile IP, handover/handoff etc.
In this context routing and forwarding is also important matters. In the end we will also show our own solution to solve mobility problems.

Preface

This thesis is a part of our graduate degree (siv.ing) in Information and Communication Technology at Agder University College in Norway. The final term of our study we have spent in Ireland as exchange masters students at Waterford Institute of Technology (WIT) in Ireland, where this thesis has been written.
We have elaborated and discussed the concepts of the Bluetooth stack, its scalability, its efficiency and its mobility. We have also proposed solutions that we believe should be taken into consideration in the future development Bluetooth.
We would like to thank our supervisor Dr Paul O Leary at Waterford Institute of Technology and Magne Arild Haglund at Agder University College for help and support while writing this thesis. We would also like to thank our friend Kjetil Wiig and Shane Dempsey at TSSG for their contributions along the way.

Håkon Gunleifsen

Kjell Rune Øyras

Index

1 Background

Objectives

- Give a short introduction to the Thesis
- Present the thesis specification.
- Name some of our challenges
- Describe what we will do in the thesis

Contents

Motivation

The last decade has been revolutionary for the Internet and associated services. This has given us the possibility to access information sources worldwide, but also exchange information in ways that were impossible a few years ago. These resources are now accessible from home, office or school. The innovation is however about to change direction due to the increasing variety of mobile devices and the fact that many people today need access to information everywhere at anytime.

In the future it is expected that the number of terminals with wireless access to network resources will increase. The present large number of mobile devices today predicts that even if only a small part of the mobile devices demanding wireless IP services, then a considerable market and important income for telecommunication operators and computer developers. It is possible that every fridge, every kitchen range and televisions are supposed to have an internet connection in the future.

As a part of this development the wireless technology Bluetooth has been considered as a short range wireless solution. We have considered the following scenario:

A central node with internet connection is standing in your house. The node has a Bluetooth interface to all devices in your house and for instance GPRS or LAN connection to the rest of the Internet.

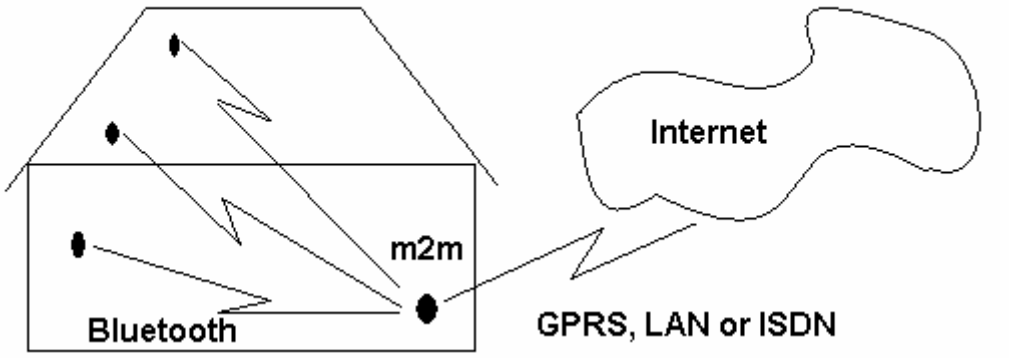


Figure 1: Scenario of the Bluetooth PAN

The different Bluetooth devices can be a PDA, a fridge or a Laptop. The central node is called a Machine-to-machine-module (M2M module) or it could also be a LAN Access point (LAP)

We predict that Machine-to-machine communication between mobile, portable or stationary devices (clients) and centralized servers will grow exponentially in the coming years. This will open for new ways to utilize the networks. To illustrate the possibilities for M2M communication, e.g. devices can be used for remote control, surveillance, tracking, localization and telemetry.
Given the low cost of the communication device, a simple connection to the equipment that should be controlled and the right network technology, M2M has the potential to become an important factor for wireless communication in the coming years. Modules can be used in all kind of wireless networks, from private networks using Bluetooth, to the cellular networks of today and tomorrow.

Thesis Specification

Our project is based on research on Machine to Machine (M2M) modules. M2M modules can be considered as gateways to the Internet, i.e. from Bluetooth to GPRS.

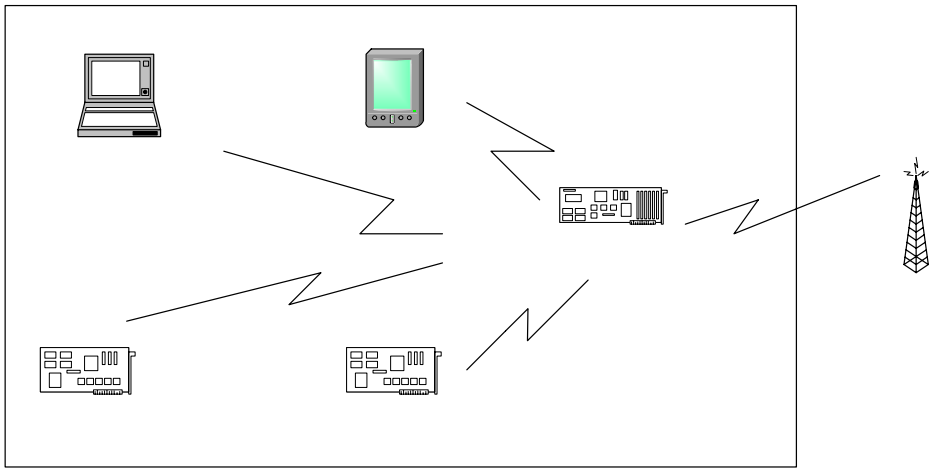


Figure 2: Bluetooth PAN with a M2M module

The M2M communications should involve IP over Bluetooth. In M2M modules, the use of IP raises many interesting questions, in particular regarding the IP-Bluetooth interface.

There have been a lot of reservations about how Bluetooth should support IP networking. Much of the discussion concerns connectionless data transmission. Some of the problems we are dealing with are IP broadcast, IP autoconfiguration, IP based Ad-hoc routing, IP mobility and IP discovery.

The implementation of IP over Bluetooth has not been specified in the Bluetooth stack, but there are several temporary solutions that more or less work.

We want to evaluate these temporary solutions and write a theoretical description of how IP over Bluetooth can work, i.e. make a point to point connection between two Bluetooth modules, using IP over Bluetooth.

This thesis is theoretical in its approach, aiming to examine the feasibility of the chosen topic.

Thesis Main Topic

Until now, bearer services for M2M applications in GSM have been the short message service and circuit switched connections. The nature of these bearers has not been optimal for the bursty traffic pattern of many M2M applications, due to the long set-up times (CS) and slow response (SMS). GPRS now adds packet-based infrastructure to GSM. With this network technology it is possible to stay connected all the time, as charging is based on volume of transferred data.
M2M modules are growing in popularity and soon we should see M2M modules with LAN, GSM/GPRS, Bluetooth, WLAN and other technologies capability.
Hence, M2M modules can operate in different modes. Sometimes they act like a LAN access point and other times as a GPRS gateway.
As we mentioned in the project specification, we will focus on the Bluetooth part of these modules. The GPRS and LAN issue will be given lower priority.

What to do?

This thesis will focus on several aspects of IP over Bluetooth. It will analyze how IP over Bluetooth is implemented today and what it probably would be like in the future. We will suggest different methods to run IP over Bluetooth and evaluate these methods. It will especially focus on one certain way to realize IP over Bluetooth, called BNEP and advantages and difficulties concerning this specific protocol.
To give a brief introduction of some of the possibilities we have in running IP over Bluetooth, we made some simplified figures.

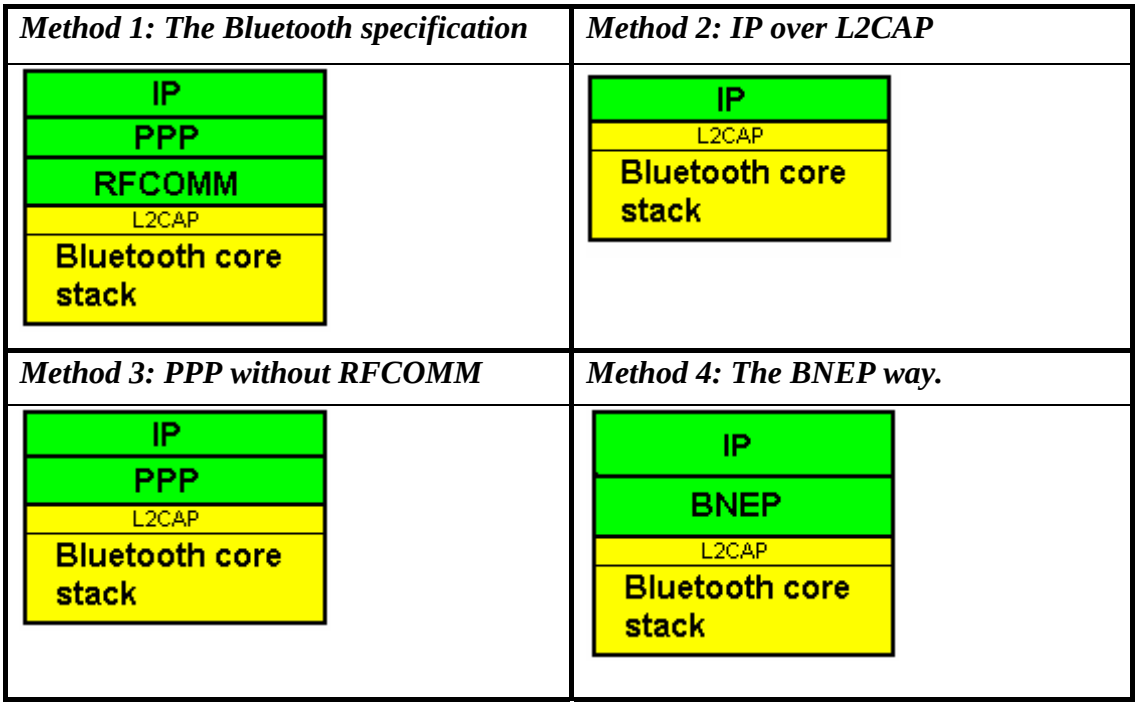


Figure 3: Different ways to run IP over Bluetooth

We choose to emphasize the first and the last of these methods. We will give an evaluation in respect of IP, routing, forwarding and the other protocol relations.

The other two methods look less likely, as PPP, L2CAP or IP would have to be changed. These protocols are so widely accepted and would be difficult to change.

Challenges

- At the moment there is a lot of work going on concerning Bluetooth and IP. In other word we felt like working in parallel with other working groups like SIG [1] Among other things the new Bluetooth Specification v2.0 is possibly being released in the end of May this year, when we are finished with our thesis. The last BNEP specification was released the 8th of May.
- Another challenge is that Bluetooth has still no support of seamless roaming or handover. Every connection has to be shut down and turned on again. The way this is solved today is through advanced use of Mobile IP.
- So far, Bluetooth is not implemented in any OS-stack, except Linux. This contributes to less acceptability and compatibility among the majority of PC users. It is expected that Microsoft and other vendors will implement Bluetooth support in their next OS-version. This will probably result in an increase in employment of Bluetooth PCMCIA cards.

Thesis Structure

In this thesis we will look at IP over Bluetooth in several contexts. We will consider different protocol stacks like we mention above and Bluetooth in a larger system, scatternets and PANs

Chapter 1:	A brief introduction to the thesis.
Chapter 2:	Description of the Bluetooth specification.
Chapter 3:	A more detailed description of the Bluetooth LAN profile and some thoughts concerning IP issues.
Chapter 4:	Evaluation of other ways to run IP over Bluetooth
Chapter 5:	Evaluation of the BNEP protocol and how it handles IP in different ways
Chapter 6:	Scatternets and how a larger scaled Bluetooth network will affect IP and discussing some roaming issues and difficulties
Chapter 7:	This chapter introduces a scenario of how we suggest making a Personal Area Network (PAN). There have been several solutions that we feel are more or less poorly designed. We want to make a cheap, relative fast and stable Bluetooth network that will be widely employed. The solution is based on the BNEP protocol.

Literature Review

WWW is a tool where the majority of the background information is obtained. When it comes to information about Bluetooth, the source of information seems inexhaustible. We have been trying to insert references marked like this: [xx] The reference you can find in the end of this report.
For general issues about Bluetooth, we have used only used one book [2]. The rest of the information is based from the Internet. We have used a lot of information from the SIG (Bluetooth Special Interest Group) [1], from other master thesis and work from other groups that we have based our work on.

2 The Bluetooth specification

Objectives

- What is Bluetooth?
- Give a introduction to the main Bluetooth Specification made by SIG
- Explanation of all the important layers in the core Bluetooth stack

Contents

Who has developed the Bluetooth stack?

Bluetooth was proposed in the Ericsson laboratories in 1994. Engineers saw a need for the wireless transmission technology that would be cheap, robust and flexible and consume little power. Technology that from one side could replace cables, but also offer new possibilities and cross the boundaries set by the existing solutions. This is why it has been called after the legendary Scandinavian king Harald Bluetooth, who christianised Denmark and united Denmark and Norway in the 9th century.

Figure 4: Bluetooth, a wireless link between devices

Ericsson Mobile Communications, Intel, IBM, Toshiba and Nokia Mobile Phones formed a Bluetooth Special Interest Group (SIG)[1] in 1998. The group's intention is to create new technology and promote it on the market. By now, almost 2500 companies have joined the SIG and work on the development and promotion of Bluetooth products.

SIG have during the past years released 2 common stacks. The latest is version 1.1. The next version is 2.0 and is expected to be finished in the beginning of this summer. The IEEE has also participated in the development of the stack. Both organizations are overlapping each other, but the figure shows how the work is divided. IEEE is focusing on the IP part of the stack, while SIG focuses on the core specification.

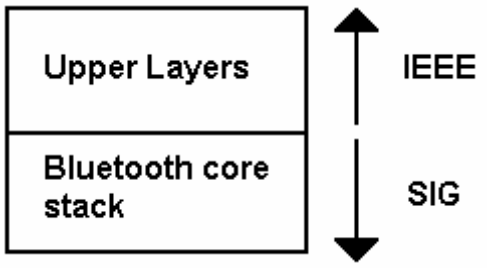


Figure 5: IEEE and SIG have work with different parts of the stack

The SIG specification is divided in two parts.

The Bluetooth Specification contains the information necessary to ensure that diverse devices supporting the Bluetooth wireless technology can communicate with each other worldwide. The specification is divided into two documents: a Core Specification (Volume I) and Profile Definitions (Volume II).

- The **Specification** describes **how the technology works** (i.e. the Bluetooth protocol architecture).
- The **Profiles** describe **how the technology is used** (i.e. how different parts of the specification can be used to fulfill a desired function for a Bluetooth device).

The Bluetooth Specification in general is introduced in this chapter. The next chapter contains a description of the LAN Profile.

According to the SIG specification this is how the Bluetooth stack is specified:

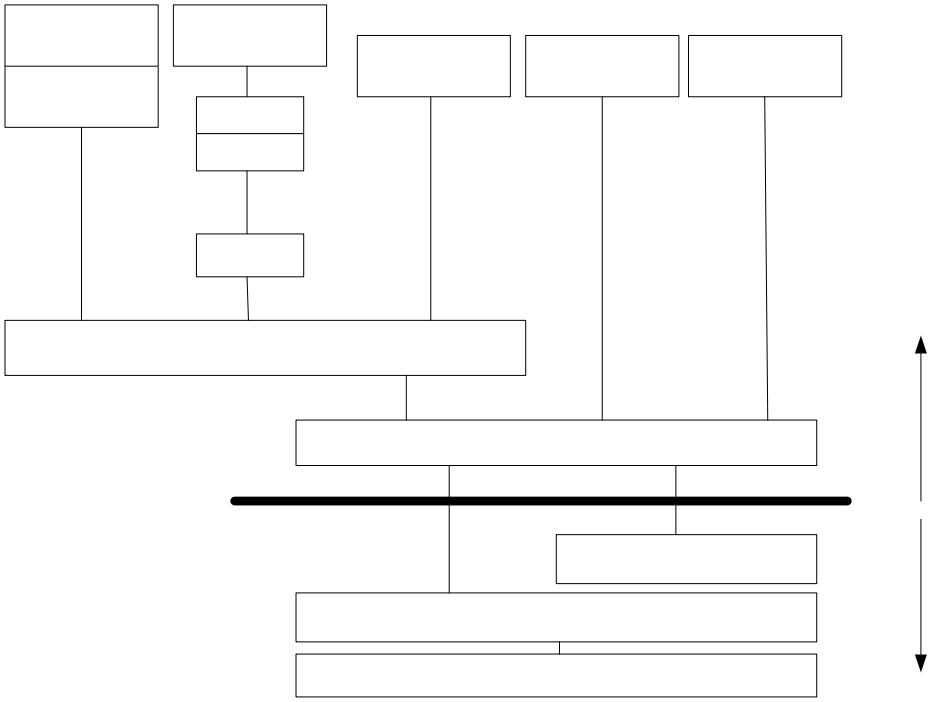
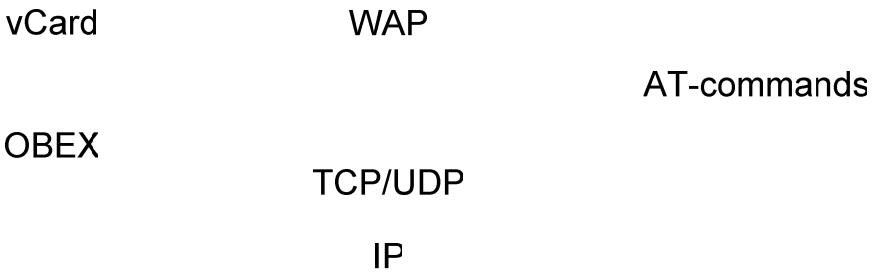


Figure 6: The Bluetooth Specification

To simplify this figure we made our own simplification of the Bluetooth stack that we will refer to in the rest of the paper (next figure).



PPP

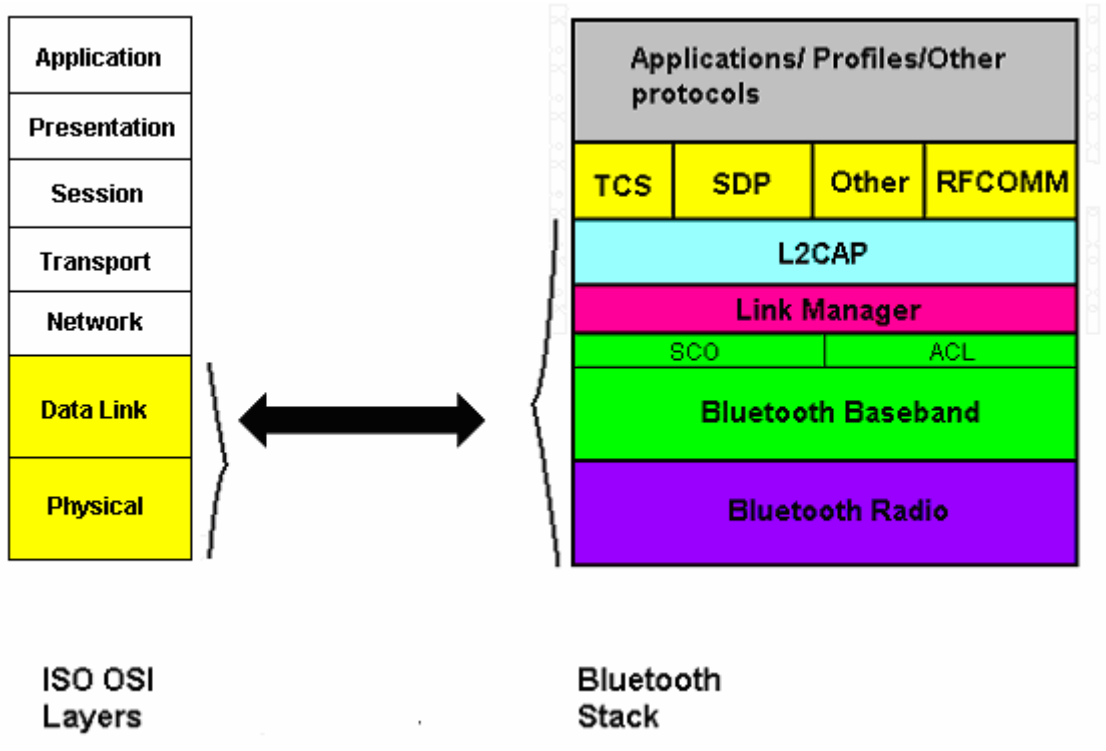


Figure 7: The simplified Bluetooth stack and how we can map it to the OSI layers.

Bluetooth Technology Overview

The principles of the Bluetooth technology are:

- Low cost: $5-10 per unit (will they ever be? Now they cost 10 times more)
- small size
- low power consumption
- speeds up to 721 kbps
- range of 10 m (extendable to100 m)

The Bluetooth system consists of a radio unit, a link control unit and a support unit for link management and host terminal interface functions. Bluetooth radio operates in the 2.4 GHz ISM (Industry, Science and Medicine) band. The range of Bluetooth radio is anywhere from 10 m. (home) to 100 m. (Airport lounge) depending on the power of the transmitter at the antenna. Depending on the class of the device, a Bluetooth radio can transmit up to 100 mW (20 dBm) to minimum of 1 mW (0 dBm) of power. It uses frequency hopping for low interference and fading, uses a TDD (Time-Division Duplex) scheme for full duplex transmission and transmits using GFSK (Gaussian Frequency Shift Keying) modulation.
The Bluetooth protocol uses a combination of circuit and packet switching. The channel is slotted and slots can be reserved for synchronous packets. The Bluetooth protocol stack can support an asynchronous connectionless (ACL) link for data and up to three simultaneous synchronous connection oriented (SCO) links for voice or a combination of asynchronous data and synchronous voice (DV packet type). Each voice channel supports a 64 Kb/s synchronous channel in each direction. The asynchronous channel can support maximum of 723.2 Kb/s uplink and 57.6 Kb/s downlink (or vice versa) or 433.9 Kb/s symmetric links. The stack primarily has a baseband for physical layer and link manager and controller for link layer. The upper layer interface depends on how these two layers are implemented and used with applications. The stack is shown below.

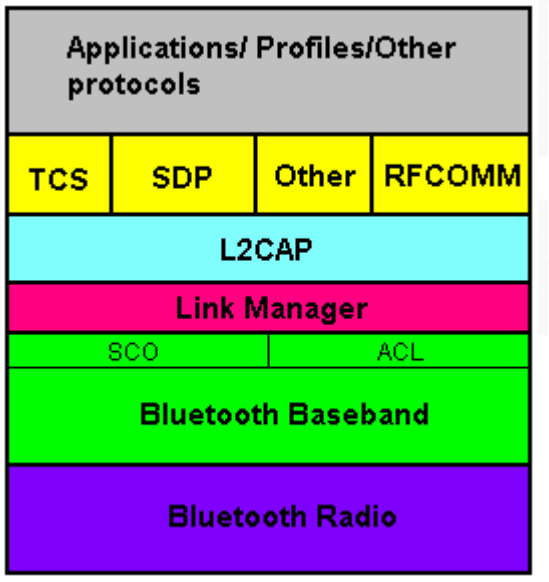


Figure 8: Simplified Bluetooth protocol stack

The stack primarily contains a physical level protocol (Baseband) and a link level protocol (LMP) with an adaptation layer (L2CAP) for upper layer protocols to interact with the lower Bluetooth stack.

Piconets and scatternet

The Bluetooth system has both a point-to-point connection and a point-to-multipoint connection. In point-to-multipoint connection, the channel is shared among several Bluetooth units. Two or more units sharing the same channel form a **piconet.** There is one master unit and seven active slave units in a piconet. These devices can be in either of the states: active, park, hold and sniff. Multiple piconets with overlapping coverage areas form a **scatternet.**

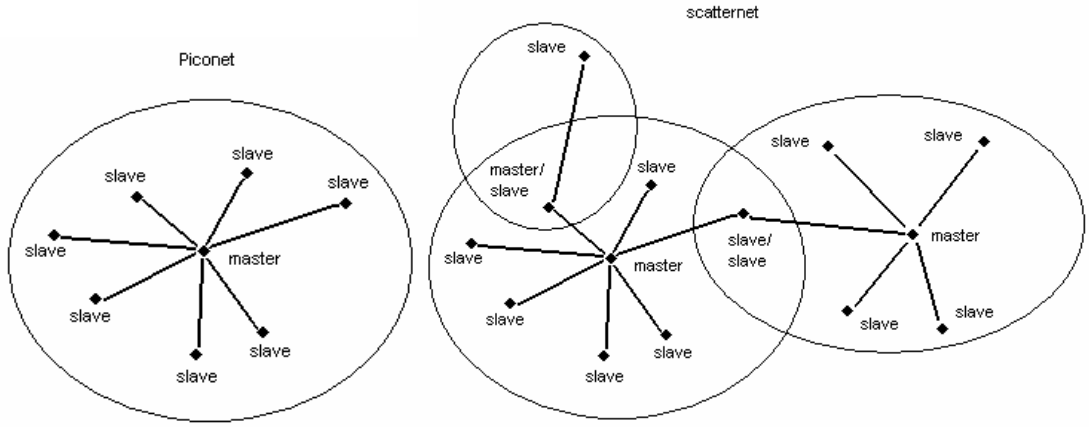


Figure 9: The difference between a piconet and a scatternet

The network in Bluetooth has a star topology. It is organized in so-called piconets. Piconets team up to 8 active devices, one of them selected as the master. A master is the device that controls the network: it chooses the hopping scheme, decides who may transmit and is used as a synchronization reference point.

Apart from active state devices may be “parked” in the piconet. The number of parked devices is only limited by the length of so-called Parked Member Address (PM_ADDR), which is 8 bits, allowing the master to keep track of up to 255 parked devices.
“Parked” state means that the device is synchronized with the master and knows the hopping sequence, but cannot transmit or receive data until it becomes “active”. This allows, for example, parked device to enter low power consumption mode and simultaneously let other devices use the bandwidth.
Several piconets with overlapping coverage areas form a scatternet. Scatternet architecture allows for transmission between devices that are not directly connected, e.g. due to the long distance. They can communicate via another device, which is in the range of both of them. All the transmission across piconets has to pass through the master device. It is important to note that the master is only piconet-local, i.e. device being master in one piconet may be slave in another piconet.

Radio interface

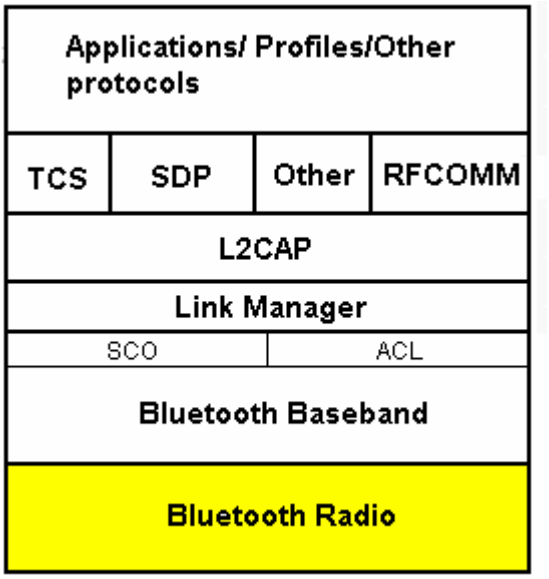


Figure 10: The Bluetooth Radio in the Bluetooth stack.

One of the most important advantages of Bluetooth is the usage of the unlicensed Industrial, Scientific & Medical (ISM) band at 2.45 GHz, occupying frequency from 2400 to 2.4835 MHz. The same band is used by WLAN technology (IEEE 802.11b) and microwave ovens.

Location	Bandwidth	RF Channels
USA, Europe and most other Countries	2400 – 2.4835 GHz	f=2402+k MHz, k=0,…,78
France	2.4465 - 2.4835	f = 2454 + k MHz, k= 0,...,22

Table 1: Bluetooth frequencies and channels

This band is divided into 79, 1 MHz wide, channels, which are used for both transmission and reception. The frequency is shared based on Time Division Duplex (TDD) technology. Due to the strict regulations existing in the ISM band and to the limiting of power transmission there are limitations in the power level. Basic transmission power is set to 0 dBm and is regulated by the Link Manager Protocol (LMP) layer.

Transmitter and Receiver requirements

The transmitter uses GFSK (Gaussian Frequency Shift Keying) where a binary one is represented by a positive frequency deviation and a binary zero by a negative frequency deviation. The definition of a Bluetooth modulated signal is given below:

Modulation	GFSK
Modulation index	0.32 +/- 1%
BT	0.5 +/- 1%
Bit Rate	1Mbps +/- 1 ppm
Modulating Data	PRBS9
Frequency accuracy better than	+/- 1 ppm

Table 2: Bluetooth Radio definition

The Bluetooth devices are classified into three power classes depending on the maximum output power of the transmitter. A power controller can be used for limiting and optimization of the output power depending on the power requirements of the device.

Power class	Maximum Output Power	Minimum Output Power
1	100 mW (20 dBm)	1 mW (0 dBm)
2	2.5 mW (4 dBm)	0.25 mW (-6 dBm)
3	1 mW (0 dBm)	N/A

Table 3: Bluetooth power classes

The actual sensitivity level is defined as the input level for which a raw bit error rate (BER) of 0.1 % is met. The requirement for a Bluetooth receiver is an actual sensitivity level of -70 dBm or better.

Bluetooth Baseband

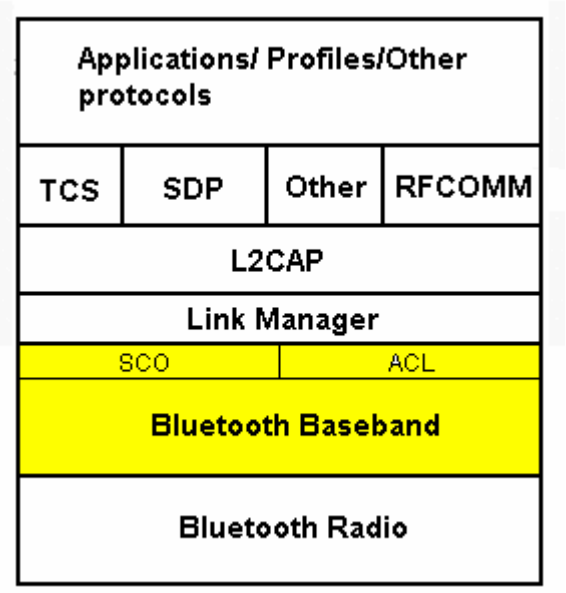


Figure 11: The Bluetooth Baseband in the Bluetooth stack.

This is the most comprehensive part of the Bluetooth protocol and one which is most important.
Baseband is the physical layer of the Bluetooth which manages physical channels and links apart from other services like error correction, data whitening, hop selection and Bluetooth security. Baseband lies on top of Bluetooth radio in the Bluetooth stack and essentially acts as a link controller and works with the link manager in carrying out link level routines suck as link connection and power control. Baseband also manages asynchronous and synchronous links, handles packets and paging, inquiry to access and inquiries to the Bluetooth devices. The Baseband transceiver applies a time-division duplex (TDD) scheme (alternate transmit and receive). Therefore apart from different hopping frequency (frequency division), the time is also slotted. In the normal connection mode, the master shall always start at even numbered slots and slave transmission shall always start at odd numbered slots (though they may continue to transmit regardless of the number of the slot).

ACL and SCO links

Baseband handles two types of links: SCO (Synchronous Connection-Oriented) and ACL (Asynchronous Connection-Less) link. The SCO link is a symmetric point-to-point link between a master and a single slave in the piconet. The master maintains the SCO link by using reserved slots at regular intervals (circuit switched type). The SCO link mainly carries voice information. The master can support up to three simultaneous SCO links while slaves can support two or three SCO links. SCO packets are never retransmitted. SCO packets are used for 64 kB/s speech transmission.
The ACL link is a point-to-multipoint link between the master and all the slaves participating on the piconet. In the slots not reserved for the SCO links, the master can establish an ACL link on a per-slot basis to any slave, including the slave already engaged in an SCO link (packet switched type). Only a single ACL link can exist. For most ACL packets, packet retransmission is applied.

Logical Channels

Bluetooth has five logical channels which can be used to transfer different types of information. LC (Control Channel) and LM (Link Manager) channels are used in the link level while UA, UI and US channels are used to carry asynchronous, isosynchronous and synchronous user information.

Bluetooth Addressing

There are basically four types of device addresses in Bluetooth:

BD_ADDR	48 bit Bluetooth device address (IEEE802 standard). It is divided into LAP (Lower Address Part of 24 bits), UAP (Upper Address Part of 8 bits) and NAP (Non-significant Address Part of 16 bits).
AM_ADDR	3 bit active member address. The all zero AM_ADDR is for broadcast messages.
PM_ADDR	8-bit member address that is assigned to parked slaves.
AR_ADDR	The access request address is used by the parked slave to determine the slave-to-master half slot in the access window it is allowed to send access messages.

Table 4: Bluetooth addresses

Bluetooth packets

The data on the piconet channel is conveyed in packets. The general packet is shown below:

Access Code(72)	Header(54)	Payload(0-2745)

Table 5: A standard Bluetooth packet (size in bits)

Access code is used for timing synchronization, offset compensation, paging and inquiry. There are three different types of Access code: Channel Access Code (CAC), Device Access Code (DAC) and Inquiry Access Code (IAC). The channel access code identifies a piconet (unique for a piconet) while DAC is used for paging and its responses. IAC is used for inquiry purposes. The header contains information for packet acknowledgement, packet numbering for out-of-order packet reordering, flow control, slave address and error check for header. The packet payload can contain either voice field, data field or both. The packet can occupy more than one slot (multi-slot packets) and can continue transmission in the next slot. The payload also carries a 16-bit CRC for error detection and correction in the payload. SCO packets do not include CRC.
There are five common types of packet, four SCO packets and seven ACL packets. See appendix.

Time Division duplex and multiple access technology

Bluetooth uses Time Division Duplex (TDD) principle to change between transmission and reception. Each time slot occupies 625 μsec, and each packet occupies up to 5 time slots (1, 3 or 5). This concept allows for a reduction in the overhead when transmitting lots of data.
For multiple access technology, distinguishing between the users, Frequency-Hopping Code Division Multiple Access (FH-CDMA) has been chosen. The principle of FH-

CDMA technology is changing the transmission/reception channels very frequently. In this case, hopping between the available 79 (23) channels occurs 1600 times/second.

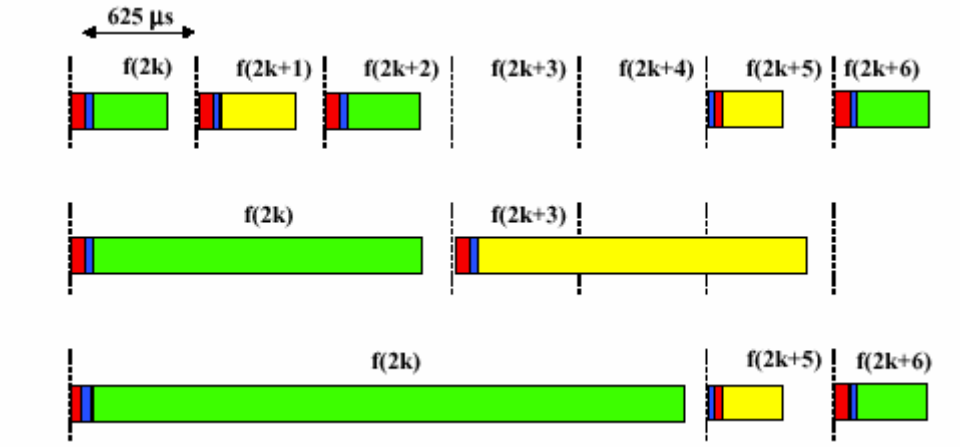


Figure 12: Division duplex and multi-slot packets

Figure 12 illustrates TDD and FH-CDMA concept. Packets sent by the master device are marked in green and slave transmission is yellow. Devices share the hopping scheme, i.e. only one may transmit on the channel at a time. 3 rows in the figure represent 3 independent hopping schemes, without overlapping frequencies

Error correction

There are three kinds of error correction schemes: 1/3 rate FEC, 2/3 rate FEC and ARQ scheme. In 1/3 rate FEC every bit is repeated three times for redundancy, in 2/3 a generator polynomial is used to encode 10 bit code to a 15 bit code, and in ARQ scheme a packet is retransmitted till an acknowledgement is received (or timeout is exceeded). Bluetooth uses fast, unnumbered acknowledgement in which it uses positive and negative acknowledgements by setting appropriate ARQN values. If the timeout is exceeded, Bluetooth flushes the packet and proceeds with the next.

Bluetooth Controller

Flow control and synchronization

Bluetooth recommends using FIFO queues in ACL and SCO links for transmission and receive. The Link Manager fills these queues and the link controller empties the queues automatically.

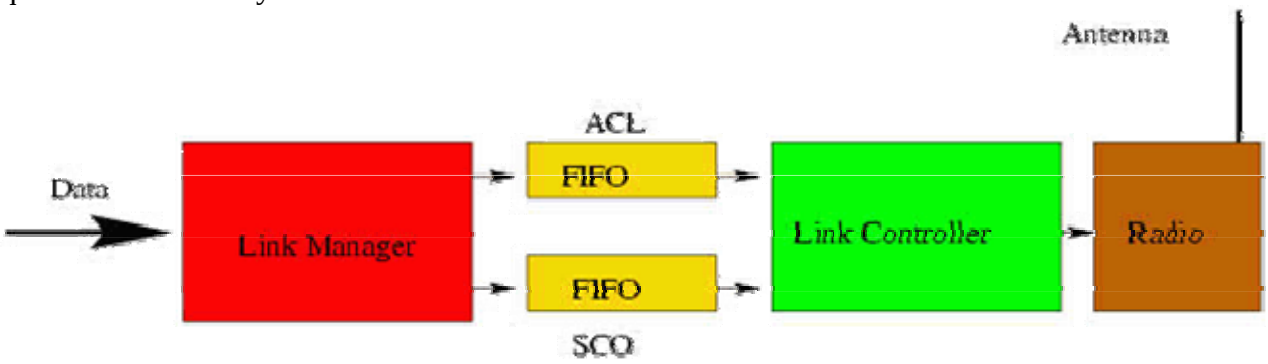


Figure 13: Flow control in Bluetooth

If these RX FIFO queues are full, flow control is used to avoid dropped packets and congestion. If data cannot be received, a STOP indication is transmitted inserted by the Link Controller of the receiver into the header of the return packet. When the transmitter receives the STOP indication, it freezes its FIFO queues. If the receiver is ready it sends a GO packet which resumes the flow again.
As mentioned earlier the Bluetooth transceiver uses a time-division duplex (TDD) scheme. This means that it alternately transmits and receives in a synchronous manner. The average timing of master packet transmission must not drift faster than 20 ppm relative to the ideal slot timing of 625 microseconds. Jitter from average timing should be less than 1 microsecond. The piconet is synchronized by the system clock of the master. The Bluetooth Device Address (BD_ADDR) of the master determines the frequency hopping sequence and the channel access code; the system clock of the master determines the phase in the hopping sequence. Master controls the traffic on the channel by a polling scheme. The master never adjusts its system clock during the existence of the piconet. The slaves adapt their native clocks with a timing offset in order to match the master clock. The Bluetooth clocks should have a resolution of 312.5 microseconds.
A 20 microsecond uncertainty window is allowed around the exact receive time in order for the access correlator for the receiver to search for the correct channel access code and get synchronized with the transmitter. When a slave returns from the hold mode, it can correlate over a bigger uncertainty window till they don’t overlap slots. A parked slave periodically wakes up to listen to beacons from the master and to resynchronize its clock offset.

Controller States

Bluetooth controller operates in two major states: *Standby* and *Connection.* There are seven sub states which are used to add slaves or make connections in the piconet. These are page, page scan, inquiry, inquiry scan, master response, slave response and inquiry response.

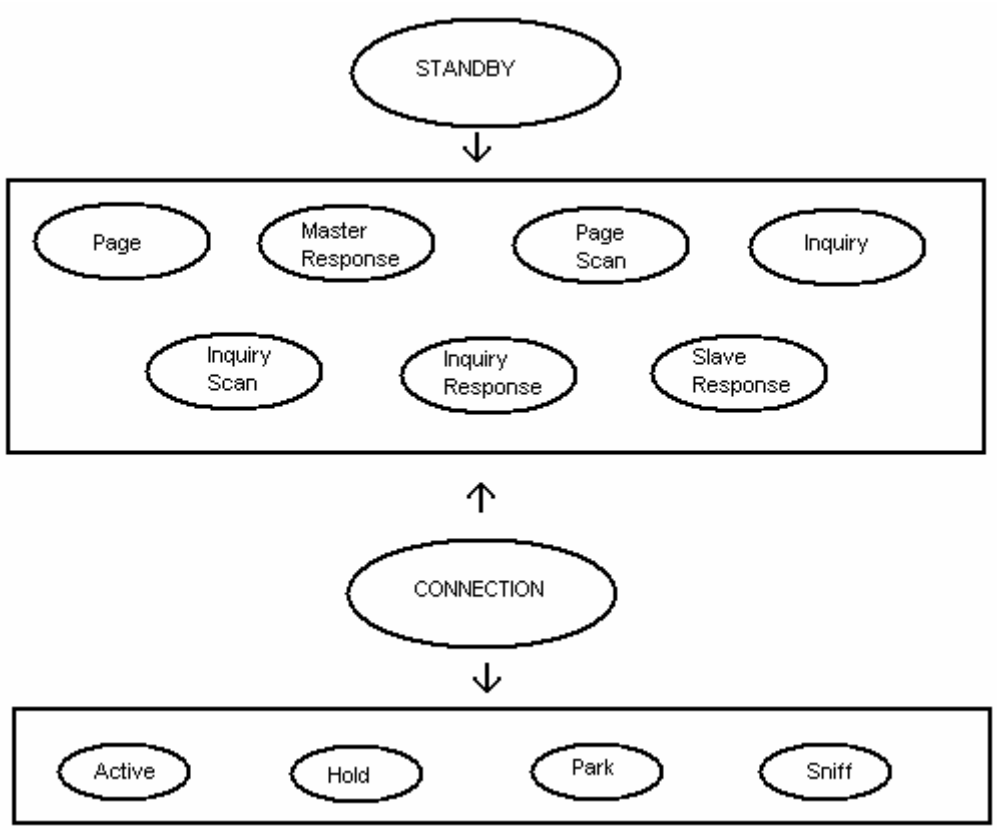


Figure 14: Bluetooth States

The *Standby* state is the default low power state in the Bluetooth unit. Only the native clock is running and there is no interaction with any device whatsoever. In the *Connection* state, the master and slave can exchange packets using the channel (master) access code and the master Bluetooth clock. The hopping scheme used is the channel hopping scheme.
Normally, a connection between two devices occurs in the following fashion. First master uses the GIAC and DIAC to inquire about the Bluetooth devices in the range (Inquire substate). If any nearby Bluetooth device is listening for these inquiries (Inquiry scan substate), it responds to the master by sending its address and clock information (FHS packet) to the master (Inquiry response substate). After sending the information, the slave may start listening for page messages from the master (Page scan). The master after discovering the in range, Bluetooth devices may page these devices (Page substate) for connection setup. The slave in page scan mode if paged by this master will respond (Slave response substate) with its device access code (DAC). The master after receiving the response from the slave may respond by transmitting the master's real time clock, master's BD_ADDR, the BCH parity bits and the class of the device (FHS packet). After slave has received this FHS packet, both enter into the *Connection* state.

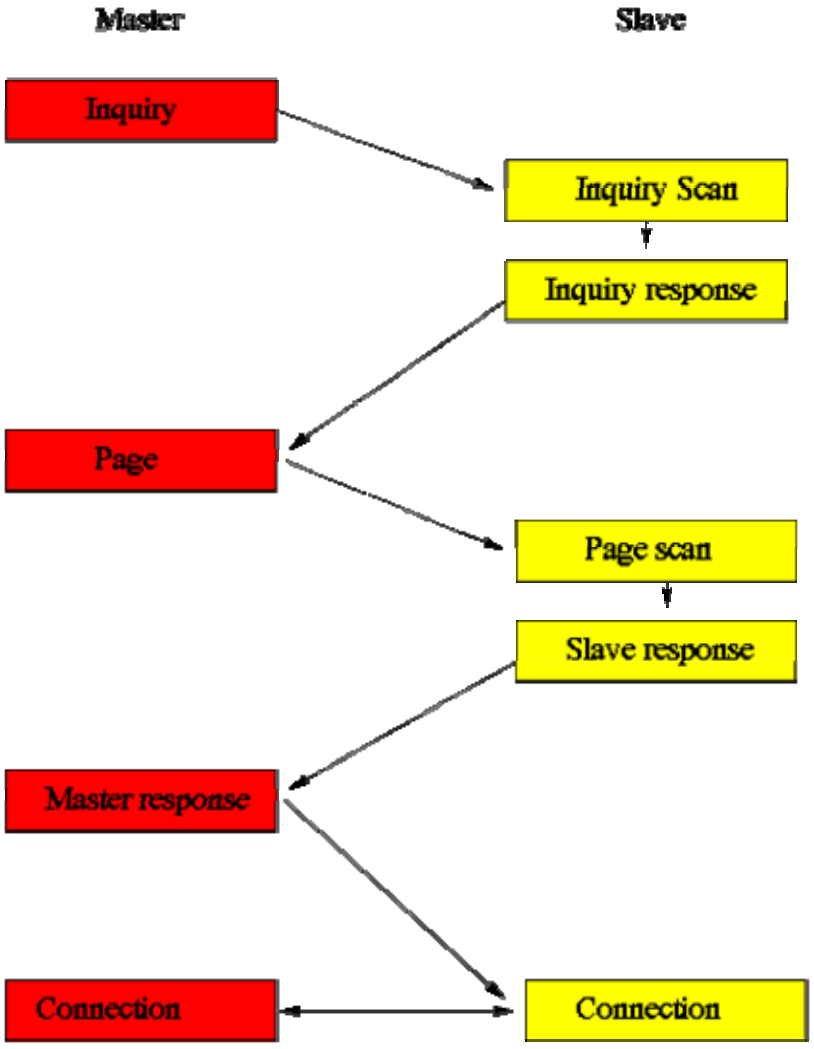


Figure 15: Bluetooth controller states

Below we describe each of these states briefly:

Page	This substate is used by the master to activate and connect to a slave. Master sends page messages by transmitting slave's device access code (DAC) in different hop channels.
Page scan	In this substate, a slave listens for its own device access code (DAC) for a duration of scan window. The slave listens at a single hop frequency (derived from its page hopping sequence) in this scan window
Slave response	The Slave responds to master's page message in this substate which is resulted if slave correlates in the page scan substate to the master's page message. The Slave enters **Connection** state after receiving FHS packet from master.
Master response	The Master reaches this substate after it receives slave's response to its page message for it, Master sends a FHS packet to the Slave and The slave replies then master enters **Connection** state.

Inquiry	Inquiry is used to find the identity of the Bluetooth devices in the close range. The discovering unit collects the Bluetooth device addresses and clocks of all units that respond to the inquiry message.
Inquiry scan	In this state, the Bluetooth devices are listening for inquiries from other devices. In this scanning device may listen for general inquiry access code (GIAC) or dedicated inquiry access codes (DIAC).
Inquiry response	For inquiry, only slave responds but not the master. The slave responds with the FHS packet which contains the slave's device access code, native clock and some other slave information.

Table 6: Bluetooth controller states

Connection States

The *Connection* state starts with a POLL packet sent by the master to verify that slave has switched to the master's timing and channel frequency hopping. The slave can respond with any type of packet.
A Bluetooth device in *Connection* state can be in any of the four following states: *Active, Hold, Sniff* and *Park* mode. One of the challenges in Bluetooth is to move between these states especially from Park to Active and vice versa. These modes are described briefly below:

Aktive	In this mode both master and slave participate actively on the channel by listening, transmitting or receiving the packets. Master and slave are kept synchronized to each other.
Sniff	In this mode slave rather than listening on every slot for master's message for that slave, sniffs on specified time slots for its messages. Hence the slave can go to sleep in the free slots thus saving power.
Hold	In this mode, a device can temporarily not support ACL packets and go to low power sleep mode to make the channel available for things like paging, scanning etc.
Park	When a slave does not need to participate on the piconet channel, but still wants to remain synchronized to the channel. It can enter park mode which is a low power mode with very little activity. The device is given a Parked Member Address (PM_ADDR) and it losses the Active Member Address (AM_ADDR).

Table 7: Bluetooth controller states

Bluetooth Security

Bluetooth security is an important issue if we have to allow keyless doors and automatic billing in super-stores. At the link layer, security is maintained by authentication of the peers and encryption of the information. For this basic security we need a public address which is unique fro each device (BD_ADDR), two secret keys (authentication keys and encryption key) and a random number generator. First a device does the authentication by issuing a challenge and the other device has to then send a response to that challenge which is based on the challenge, it's BD_ADDR and a link key shared between them. After authentication, encryption may be used to enhance security.

There are four types of link keys: *combination key, unit key, temporary key* and *initialization key.*

Summary of Baseband

The Baseband protocol forms the lowest layer in Bluetooth architecture. It is responsible for the functionality contained in the physical layer of the OSI/ISO model, but also performs some tasks from higher layers. Its main tasks are:

- synchronization,
- transmission of the information,
- error correction (FEC & CRC),
- logical channels division,
- Data whitening (scrambling).

Bluetooth supports both synchronous and asynchronous channels. The possible Configurations are:

- 1 asynchronous data channel
- up to 3 simultaneous synchronous voice channels each 64kbps
- channel which simultaneously supports asynchronous data and synchronous voice.

The asynchronous channel may support up to 721kbps downlink and 57.6kbps uplink or symmetrically 433.9 kbps in both directions.

Link Management Protocol (LMP)

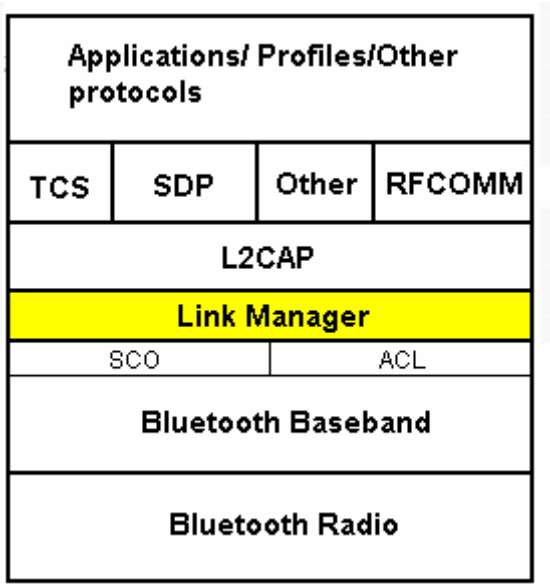


Figure 16: The Link Manager in the Bluetooth stack

Link Manager is used for managing security, link set-up and control. It talks to the other link managers to exchange information and control messages through the link controller using some predefined link-level commands. Its support for upper layer protocols is bit hazy but possibly an upper layer interface can be used to execute algorithms for mode management (park, hold, sniff, active), security management, QoS management etc. If the security is not a big issue, a user can decide about the level of security by choosing some reduced security option and therefore inform link manager to go soft on security.

Information Exchange and Request

A Bluetooth link manager can request from other link managers the clock offset (master requesting the slave to tell it the current clock offset stored by it which slave itself got from master during some packet exchange), slot offset (slot offset is the time in microseconds between the start of the master's transmission slot in the piconet where the PDU is transmitted and the start of the master's transmission slot where the BD_ADDR device in the PDU is master. It is useful in master-slave switch and inter-piconet communications), timing accuracy (clock drift and jitter), link manager version and information about supported features like support for authentication, SCO packets etc.

Mode management and SCO connections

The link manager also handles master-slave switch procedure and mode switching procedures (forcing or requesting a device to change mode to either hold, sniff or park mode). In parking mode, it has to take care of how to broadcast a message to the parked devices, how to handle beacon parameters and how to unpark a parked device gracefully.

Apart from above the features a link manager can handle power control (lower or increase the power) and can establish SCO links by reserving slots and negotiating SCO parameters. If a device wants to establish connections using layers above the Link Manager, then it can open connections between the two devices too.

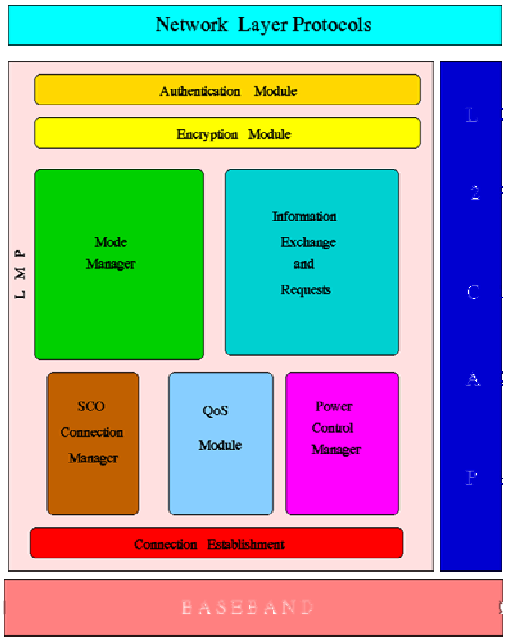


Figure 17: LMP Roles

Host Controller Interface (HCI)

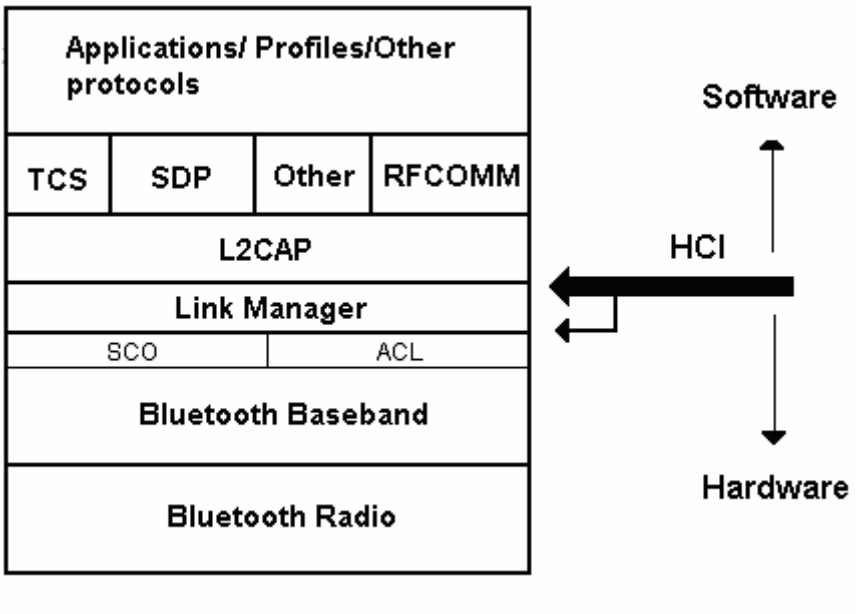


Figure 18: The HCI Interface in the Bluetooth stack

Host control interface (HCI), defines the standard interface-independent method to communicate with the Bluetooth chip. It provides a command interface to the baseband controller and link manager, and access to hardware status and control registers. Essentially this interface provides a uniform method of accessing the Bluetooth baseband capabilities. The HCI exists across 3 sections;

The Host, The Transport Layer and The Host Controller.

Each of the sections has a different role to play in the HCI system.

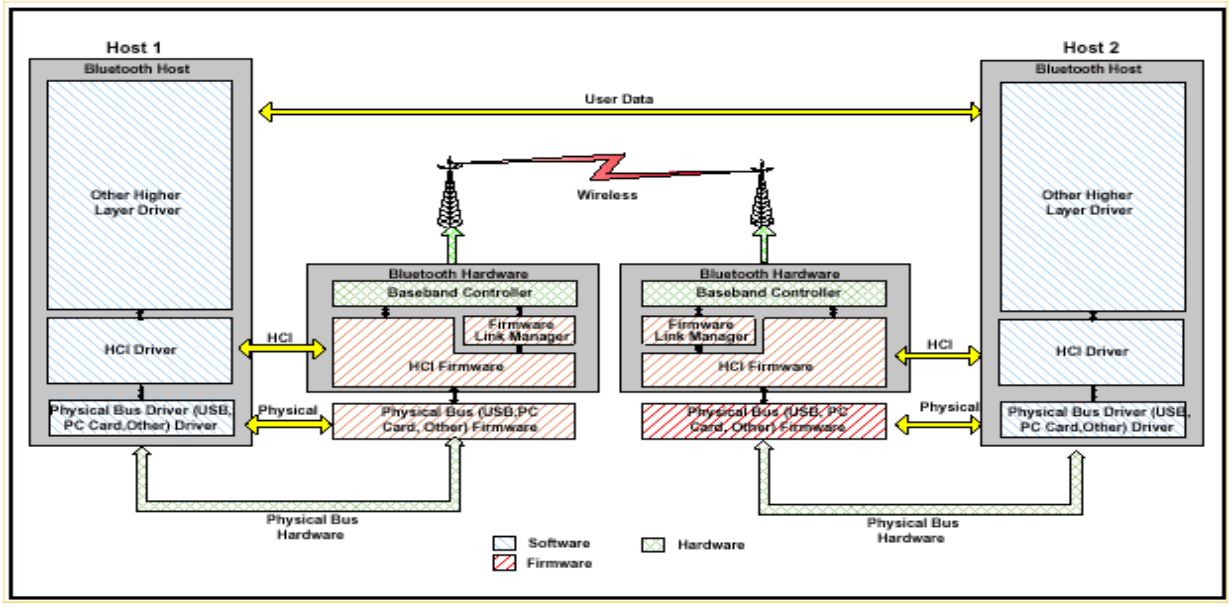


Figure 19: HCI roles

a) HCI Firmware (location: Host Controller)
HCI Firmware is located on the Host Controller, (e.g. the actual Bluetooth hardware device). The HCI firmware implements the HCI Commands for the Bluetooth hardware by accessing baseband commands, link manager commands, hardware status registers, control registers, and event registers. The term Host Controller means the HCI-enabled Bluetooth device

b) HCI Driver (location: Host)
The HCI Driver which is located on the Host (e.g. software entity). The Host will receive asynchronous notifications of HCI events, HCI events are used for notifying the Host when something occurs. When the Host discovers that an event has occurred it will then parse the received event packet to determine which event occurred. The term Host means the HCI-enabled Software Unit.

c) Host Controller Transport Layer (location: Intermediate Layers)
The HCI Driver and Firmware communicate via the Host Controller Transport Layer, i.e. a definition of the several layers that may exist between the HCI driver on the host system and the HCI firmware in the Bluetooth hardware. These intermediate layers, the Host Controller Transport Layer, should provide the ability to transfer data without intimate knowledge of the data being transferred. Several different Host Controller Layers can be used, of which 3 have been defined initially for Bluetooth: USB, UART and RS232. The Host should receive asynchronous notifications of HCI events independent of which Host Controller Transport Layer is used.

HCI Commands

The HCI provides a uniform command method of accessing the Bluetooth hardware capabilities. The HCI Link commands provide the Host with the ability to control the link layer connections to other Bluetooth devices. These commands typically involve the Link Manager to exchange LMP commands with remote Bluetooth devices. The HCI Policy commands are used to affect the behavior of the local and remote LM. These Policy commands provide the Host with methods of influencing how the LM manages the piconet. The *Host Controller and Baseband commands, Informational commands,* and *Status commands* provide the Host access to various registers in the Host Controller.

HCI-Specific Information Exchange
The Host Controller Transport Layer provides transparent exchange of HCI-specific information. These transporting mechanisms provide the ability for the Host to send HCI commands, ACL data, and SCO data to the Host Controller. These transport mechanisms also provide the ability for the Host to receive HCI events, ACL data, and SCO data from the Host Controller. Since the Host Controller Transport Layer provides transparent exchange of HCI-specific information, the HCI specification specifies the format of the commands, events, and data exchange between the Host and the Host Controller.

Link Control Commands
The Link Control commands allow the Host Controller to control connections to other Bluetooth devices. When the Link Control commands are used, the Link Manager (LM) controls how the Bluetooth piconets and scatternets are established and maintained. These commands instruct the LM to create and modify link layer

connections with Bluetooth remote devices, perform Inquiries of other Bluetooth devices in range, and other LMP commands.

Link Policy Commands

The Link Policy Commands provide methods for the Host to affect how the Link Manager manages the piconet. When Link Policy Commands are used, the LM still controls how Bluetooth piconets and scatternets are established and maintained, depending on adjustable policy parameters. These policy commands modify the Link Manager behavior that can result in changes to the link layer connections with Bluetooth remote devices.

Host Controller & Baseband Commands

The Host Controller & Baseband Commands provide access and control to various capabilities of the Bluetooth hardware. These parameters provide control of Bluetooth devices and of the capabilities of the Host Controller, Link Manager, and Baseband. The host device can use these commands to modify the behavior of the local device.

Informational Parameters

The Informational Parameters are fixed by the manufacturer of the Bluetooth hardware. These parameters provide information about the Bluetooth device and the capabilities of the Host Controller, Link Manager, and Baseband. The host device cannot modify any of these parameters.

Status Parameters

The Host Controller modifies all status parameters. These parameters provide information about the current state of the Host Controller, Link Manager, and Baseband. The host device cannot modify any of these parameters other than to reset certain specific parameters.

Testing Commands

The Testing commands are used to provide the ability to test various functionality's of the Bluetooth hardware. These commands provide the ability to arrange various conditions for testing.

Link Layer Control and Adaptation layer Protocol (L2CAP)

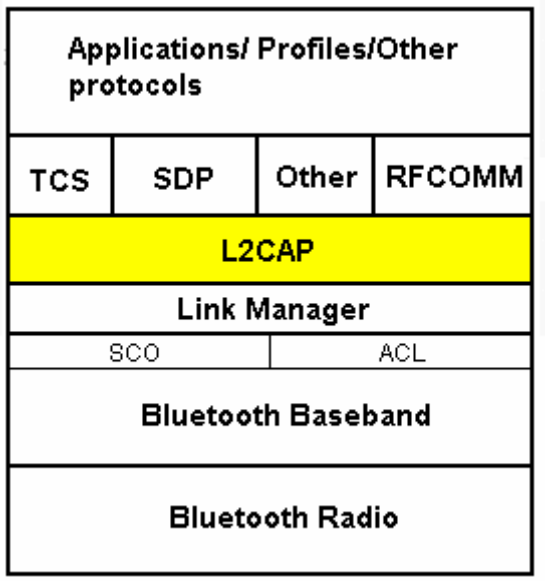


Figure 20: The L2CAP layer in the Bluetooth stack

L2CAP has many important tasks in the Bluetooth protocol stack. It is the interface to all higher layer protocols like RFCOMM, TCS and SDP. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. L2CAP permits higher level protocols and applications to transmit and receive L2CAP packets up to 64 Kilobytes in length. L2CAP only supports ACL links. L2CAP uses the concept of channels to establish different pathways between different applications on Bluetooth devices. These channels are identified by Channel Identifiers (CIDs) which represent a logical end point of a connection for each application on a device. CIDs are 16 bit numbers of which 0x0001 to 0x003F are reserved for specific L2CAP functions (0x0001 is a signaling channel, 0x0002 is a connectionless reception channel and rest are reserved or prohibited).

Connection Identifiers (CIDs)

The idea behind L2CAP is to provide an interface similar to TCP/IP function calls. In NSBLUE, the L2CAP connections and data are sent in the following manner:

```
cid = l2cap->openL2CAPConnection(ui->getContext());
l2cap->send(cid,data,len);
l2cap->recv(cid,data,len);
```

where cid is the channel identifier, data is the pointer to “char” and “len” is the length of the data. “getContext() returns the local context like “Haakon”, “Waterford” etc. “CID” is assigned from a pool of free CIDs (pool can be assigned in blocks to save memory).

Protocol Multiplexing

L2CAP also does protocol multiplexing by using the PSM field in the L2CAP Connection Request command. L2CAP can multiplex connection requests to upper layer protocols like the Service Discovery Protocol, RFCOMM and Telephony Control.

Segmentation and Reassembly

Segmentation and Reassembly (SAR) operations are used to improve efficiency by supporting a maximum transmission unit (MTU) size larger than the largest Baseband packet. This reduces the overhead by spreading the network and transport packets used by higher layer protocols over several baseband packets. L2CAP segments higher layer packets into pieces that can be passed to the Link Manager for transmission and reassembles those chunks into L2CAP packets using information provided through HCI and from the packet header. SAR is implemented using very little overhead in Baseband packets. The two L_CH bits defined in the first byte of Baseband payload (also called the frame header) are used to signal the start and continuation of L2CAP packets (L_CH shall be 10 for the first segment and 01 for a continuation segment. To avoid any reassembly problems due to out of order packets, all L2CAP segments associated with an L2CAP packet must be passed through to the Baseband before any other L2CAP packet destined to the same unit may be sent. Also Stop and Wait protocol used by Baseband makes sure that a packet is received correctly by the other unit before the next packet is sent. This avoids most of the out of the order packets, as received in wireline TCP/IP connections because of window based transmissions.

L2CAP Events and Actions

L2CAP operates using events and commands which it receives or transmits from/to upper or lower layers. These events can be, for instance, a connection request from the upper layer, a data write request or may be a disconnection request. The lower layers can tell L2CAP about incoming connection, disconnection or other requests. If L2CAP of this unit needs to talk to the L2CAP on the other unit, then it uses some special commands which are called signaling commands. These commands are generally used to establish connection-oriented channels after a link level connection is created or present. L2CAP has seven operational states: CLOSED, W4_L2CA_CONNECT_RSP, W4_L2CAP_CONNECT_RSP, CONFIG, OPEN, W4_L2CAP_DISCONNECT_RSP, and W4_L2CA_DISCONNECT_RSP. These states further make L2CAP connections look similar to TCP connections.

RFCOMM

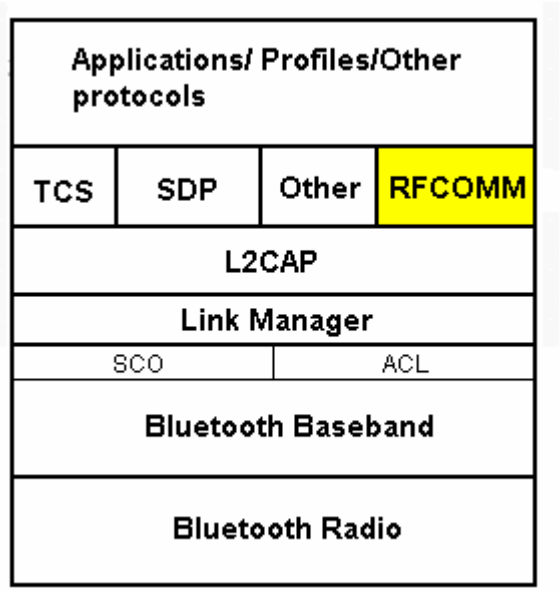


Figure 21: The RFCOMM in the Bluetooth Stack

RFCOMM is based on the ETSI TS 07.10 specification. It is a protocol used for emulation of the serial ports over L2CAP. It supports up to 60 simultaneous connections between 2 Bluetooth devices. For compatibility issues, emulation of the 9-circuit RS-232 port is supported. ETSI TS 07.10 specification has been chosen because of its flexibility and large number of applications. Among them we have modems, object exchange (OBEx) and TCP/IP stack via the PPP protocol. We will also come back to this protocol later, when we are discussing IP over RFCOMM.

Device Types

Basically two device types exist that RFCOMM must accommodate.

Type 1 Devices are communication end points such as computers and printers.
Type 2 Devices are those that are part of the communication segment; e.g. modems.

Though RFCOMM does not make a distinction between these two device types in the protocol, accommodating both types of devices impacts the RFCOMM protocol.

Control Signals

RFCOMM emulates the 9 circuits of an RS-232 interface. The circuits are listed below.

Pin Circuit Name
102 Signal Common
103 Transmit Data (TD)
104 Received Data (RD)
105 Request to Send (RTS)
106 Clear to Send (CTS)
107 Data Set Ready (DSR)
108 Data Terminal Ready (DTR)
109 Data Carrier Detect (CD)
125 Ring Indicator (RI)

Table 8: RFCOMM control signals

Multiple Emulated Serial Ports

Two BT devices using RFCOMM in their communication may open multiple emulated serial ports. RFCOMM supports up to 60 open emulated ports; however the number of ports that can be used in a device is implementation-specific. A *Data Link Connection Identifier (DLCI)* identifies an ongoing connection between a client and a server application. The DLCI is represented by 6 bits, but its usable value range is 2…61. The DLCI is unique within one RFCOMM session between two devices.
To account for the fact that both client and server applications may reside on both sides of an RFCOMM session, with clients on either side making connections independent of each other, the *DLCI value space is divided* between the two communicating devices using the concept of RFCOMM server channels.
If a BT device supports multiple emulated serial ports and the connections are allowed to have endpoints in different BT devices, then the RFCOMM entity must be able to run multiple multiplexer sessions. Note that each multiplexer session is using its own L2CAP channel ID (CID). The ability to run multiple sessions of the multiplexer is optional for RFCOMM.

Flow Control Methods

Wired ports commonly use flow control such as RTS/CTS (Ready and Clear to send) to control communications. On the other hand, the flow control between RFCOMM and the lower layer L2CAP depends on the service interface supported by the implementation. In addition RFCOMM has its own flow control mechanisms. These methods can be found in the specification. [4]

Service Discovery Protocol (SDP)

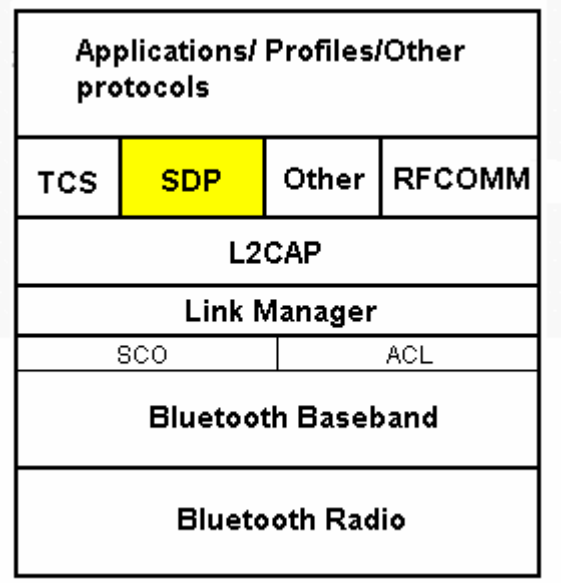


Figure 22: The SDP protocol in the Bluetooth stack

SDP is used to locate the services available in the neighboring Bluetooth devices. It can be used to discover a printer, a camera, wireless headset or any other device meeting the possessing the required properties.

SDP uses a request/response model where each transaction consists of one request protocol data unit (PDU) and one response PDU. In the case where SDP is used with the Bluetooth L2CAP transport protocol, only one SDP request PDU per connection to a given SDP server may be outstanding at a given instant. In other words, a client must receive a response to each request before issuing another request on the same L2CAP connection. Limiting SDP to sending one unacknowledged request PDU provides a simple form of flow control.

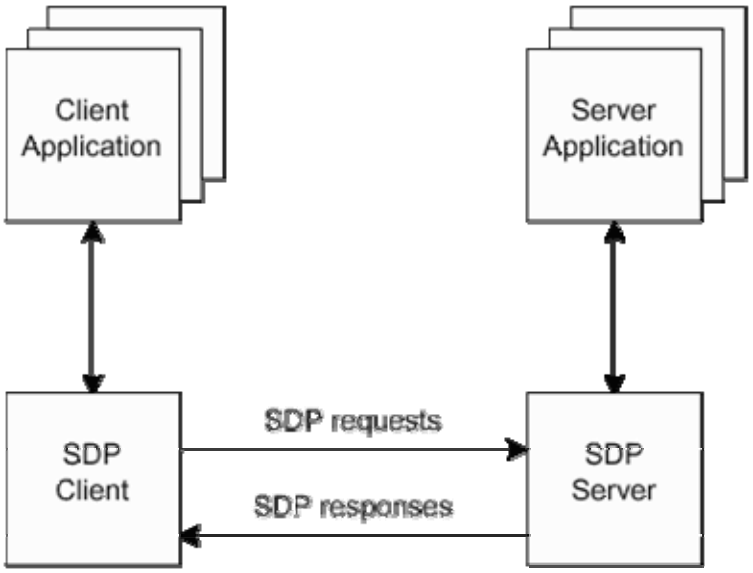


Figure 23: SDP uses request/reply messages

PDU Format

Every SDP PDU consists of a PDU header followed by PDU-specific parameters. The header contains three fields:

- **PDU ID** field identifies the type of PDU. I.e. its meaning and the specific parameters.
- **TransactionID** field uniquely identifies request PDUs and is used to match response PDUs to request PDUs.
- **Parameter Length field** specifies the length (in bytes) of all parameters contained in the PDU.

SDP Services

A Service is any enity that can provide information or perform an action. Information that is maintained by a SDP server is contained in a service record. The service record contains a list of service attributes. Service attributes describes the service and consists of two values: attribute ID and attribute value.
Each service is an instance of a service class. A service class contains all service records and is identified by a service class identifier; UUID.

Service Discovery

SDP focuses primarily on discovering services available from or through Bluetooth devices. SDP does not define methods for accessing services; once services are discovered with SDP, they can be accessed in various ways, depending upon the service.
SDP allows discovering services in various means; Searching and Browsing. In other words looking for a specific device or look at all devices.

Future extensions

Although Bluetooth has many advantages, there are some bottlenecks included in the technology concept. These shall be removed in the second release of specifications. The improvements include:

- increasing the transmission speed to 12 Mbps,
- allowing slave-to-slave communication, optimizing the connections.
- Improve Bluetooth to include roaming capability.

3 The LAN Access Profile

Objectives

- Give a short introduction to the Bluetooth Profiles
- Present the LAN Access Profile
- How to connect to a LAN access point.
- An evaluation of the running IP over Bluetooth with the LAN Access profile.
- IP issues in this context, like routing, PPP and broadcast problems

Contents

The Bluetooth Profiles

Profiles have been developed in order to describe how implementations of user models are to be accomplished. The user models describe a number of user scenarios where Bluetooth performs the radio transmission. A profile can be described as a vertical slice through the protocol stack. It defines options in each protocol that are mandatory for the profile. It also defines parameter ranges for each protocol. The profile concept is used to decrease the risk of interoperability problems between different manufacturers' products.

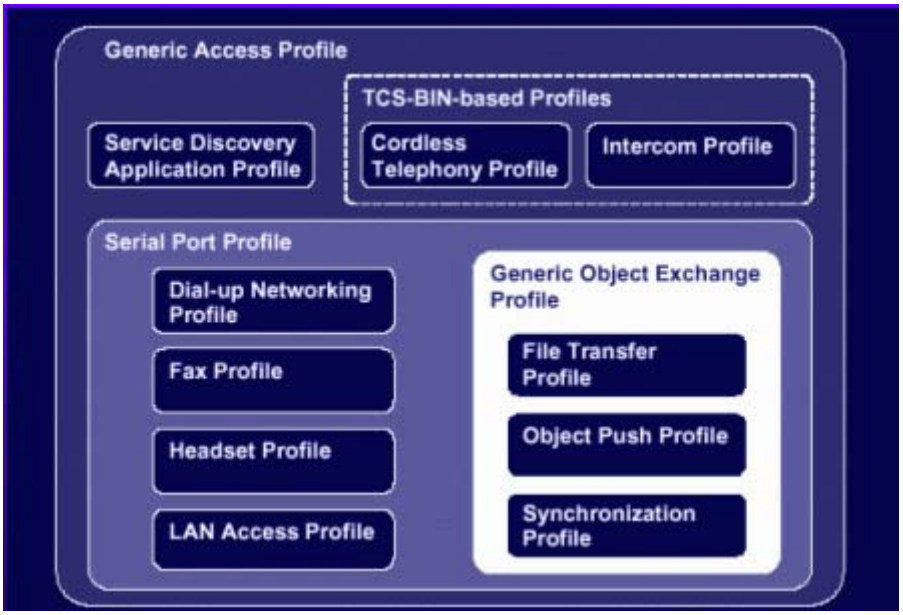


Figure 24: Bluetooth Profiles

The Bluetooth profile structure and the dependencies of the profiles are depicted above. A profile is dependent upon another profile if it re-uses parts of that profile, by implicitly or explicitly referencing it. Dependency is illustrated in the figure: a profile has dependencies on the profile(s) in which it is contained – directly and indirectly. For example, the Object Push profile is dependent on Generic Object Exchange, Serial Port, and Generic Access Profiles.

In respect of our thesis, the relevant profile is the LAN Access Profile, Serial-Port Profile and the Generic Access Profile. The focus will be on the LAN access profile.

The LAN Access Profile

The LAN Access Profile defines how Bluetooth enabled devices can access the services of a LAN using PPP. Also, this profile shows how the same PPP mechanisms are used to form a network consisting of two Bluetooth-enabled devices. Basically this profile defines LAN Access using PPP over RFCOMM.
This profile defines how PPP networking is supported in the following situations.

- LAN Access for a single Bluetooth device.
- LAN Access for multiple Bluetooth devices.
- PC to PC (using PPP networking over serial cable emulation).

PPP is the IETF Point-to-Point Protocol[5]. PPP-Networking is the means of taking IP packets to/from the PPP layer and placing them onto the LAN.

The content of this chapter will be how to realize IP over PPP over RFCOMM the way it is specified in the specification.

This is the first evaluation of the possibilities we mentioned in the beginning.

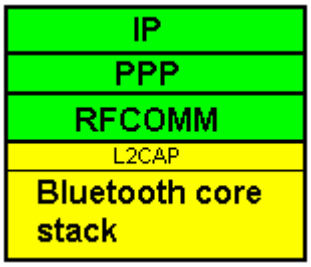
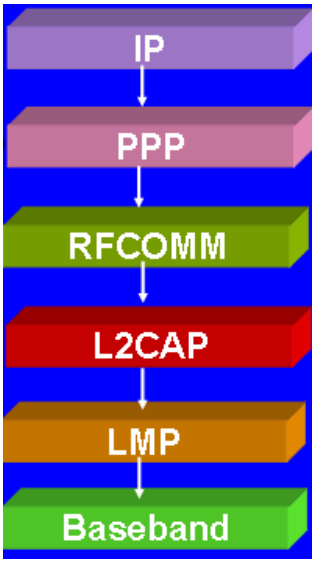


Figure 25: IP over Bluetooth according to the LAN access Profile

The LAN access profile defined in the Bluetooth specification

In the Bluetooth 1.1 specification LAN access was defined by using PPP over RFCOMM.



As the figure shows the LAN access profile specifies using PPP over RFCOMM to link an IP stack to the Bluetooth stack.

Three scenarios are covered by this profile, :

- **LAN Access by a Single PC**: One device connected to a LAN Access point. The device will simulate Dial-up networking to the LAP.
- **LAN Access by Multiple DTs**: Several devices connected to a LAN Access point. The devices will simulate Dial-up networking to the LAP.
- **PC to PC Connection:** Ad hoc networking where one device simulates a LAP.

The figure below shows a LAN access point (LAP) acting as an intermediary between a Bluetooth device and a server resident on a LAN. However, the LAN access profile simply specifies how to layer an IP stack on top of a Bluetooth stack, so it could also be used to provide a TCP/IP link between a pair of PCs or to provide a network between a group of local PCs and a LAN access point.

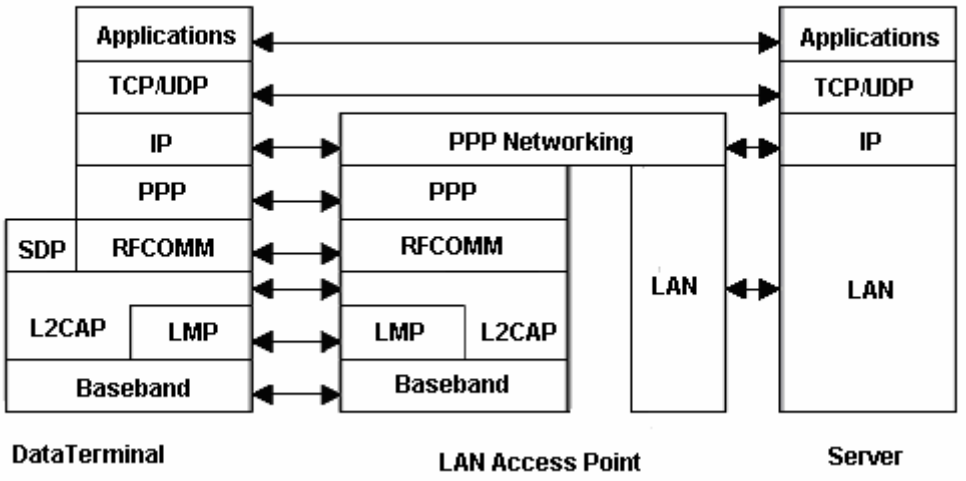


Figure 26: A connection to a LAN Access Point using the LAN profile

LAN Access Point (LAP)

This is the Bluetooth device that provides access to a LAN (e.g. Ethernet, Token Ring, Fiber Channel, Cable Modem, Firewire, USB, Home Networking, GPRS). The LAP provides the services of a PPP Server. The PPP connection is carried over RFCOMM. RFCOMM is used to transport the PPP packets and it can also be used for flow control of the PPP data stream.

Data Terminal (DT)

This is the device that uses the services of the LAP. Typical devices acting as data terminals are laptops, notebooks, desktop PCs and PDAs. The DT is a PPP client. It forms a PPP connection with a LAP in order to gain access to a LAN.

Connecting to a LAN Access Point

To reduce connection times, the LAN access point's address can be pre-entered into connecting devices. The LAN access point periodically holds its links and page scans allowing connecting devices to simply page the access point and conduct a master/slave switch to hand over control of the link to the LAN access point. It is important that the access point always becomes the master and that every slave that wants to be a master switches to be a slave. If not, the connecting device will remain as a master and of the link and the LAP would lose control over its own bandwidth and possibly preventing it from meeting QoS commitments.

The steps in connecting to a LAN access point are shown in the figure below.

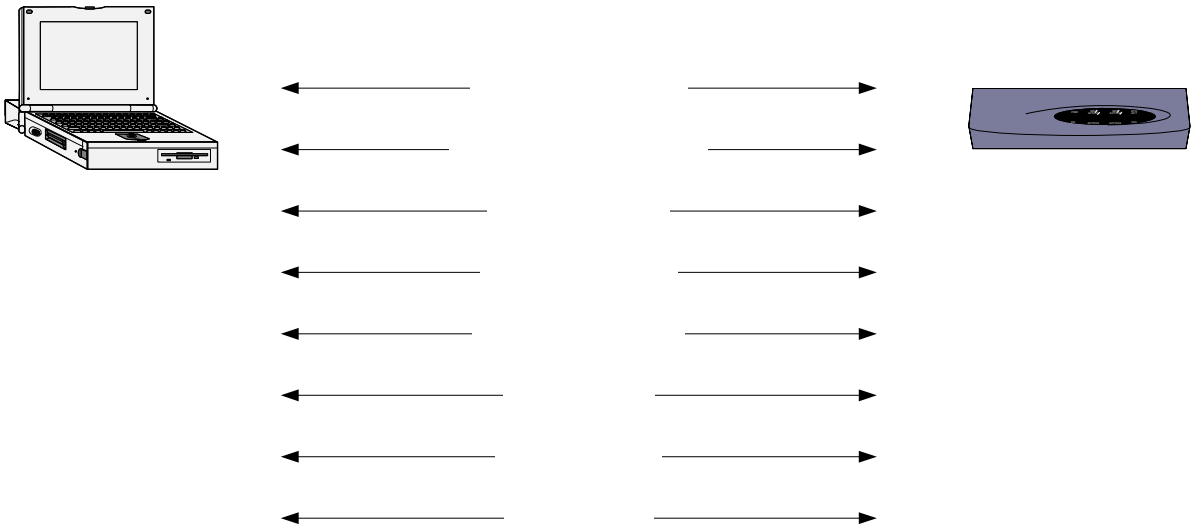


Figure 27: How to connect to a LAN access Point

First, the data terminal must find a LAN access point. It does this by inquiring (See Baseband, Chapter 2) and finding devices with the LAN access point device type. For LAN access points, the minor device type tells the data terminal how fully utilized a LAN access point is. This allows the data terminal to ignore LAN access points that are 100% utilized. If the data terminal has a choice of LAN access points to connect with, the information in the minor device type allows it to choose the least utilized one, as it is likely to deliver a better bandwidth on the link.
When the data terminal pages the LAN access point, the link will only be accepted if the data terminal is willing to do a Master/Slave switch, allowing the LAN access point to become a master.
Once a baseband link is established, an L2CAP connection is set up and the data terminal queries the service records with a service called LAN_Access_Using_PPP. This gives the parameters needed to connect to the LAN access point.
The L2CAP link used for service discovery can be dropped, and a separate L2CAP link is set up for LAN access. RFCOMM and PPP connections are established across this L2CAP link. The data terminal must then negotiate an IP address with the LAN access point using PPP. Once the data terminal has obtained an IP address, it can begin accessing the LAN. All PPP traffic is exchanged on an encryption link, but authentication at the PPP level is optional.

Data terminal

Termina
discover LA

Terminal page
and estal

LMP sets
switch

L2CAP lin
reco

Term
PPP/RFCO

PPP a
(c

Description of the involved protocols.

We are now going to evaluate the different protocols and the way they collaborate. In the end we will see different ways to run IP with RFCOMM and PPP evaluate how IP works over these protocols.

RFCOMM - The Serial Port Profile.

The RFCOMM is actually the most important part of the Serial port profile. The serial port profile provides serial cable emulation for Bluetooth devices, in this way applications do not have to be modified to use Bluetooth, they can simply treat Bluetooth as a serial cable link. The serial port profile is based on the GSM standard TS 07.10, which allows multiplexing of numerous serial connections over one serial link.

This protocol was already discussed in the second chapter when we introduced Bluetooth in general. A short summary is given here of why and how it is used in this context.
RFCOMM provides serial port emulation, which can be used to connect to legacy applications. It is also used for data transfers in several of the Bluetooth profiles, e.g. LAN profile.
RFCOMM provides multiple concurrent connections by relying on L2CAP to handle multiplexing over single connections, and to provide connections to multiple devices.
RFCOMM is a simple, reliable transport protocol with framing and multiplexing.
Because RFCOMM frames are carried in the payload of L2CAP packets, an L2CAP connection must be set up before an RFCOMM connection can be set up.
RFCOMM has a reserved Protocol and Service Multiplexer value which is used by L2CAP to identify RFCOMM traffic. Any L2CAP frames received with this value in the PSM field will be sent to RFCOMM for processing.
To set up an RFCOMM connection, an L2CAP connection must first be set up. RFCOMM frames are sent in the payload field of L2CAP packets. Once the L2CAP connection is set up, RFCOMM control frames are sent back and forth to establish a signaling channel with a Data Link Connection Identifier (DLCI) set to 0. After this is set up, subsequent channels are established for transferring data. Up to 30 data channels can be set up, so RFCOMM can theoretically support 30 different services at once. (In practice, most Bluetooth devices will not have the resources to support 30 different resources).

PPP and PPP LINKS

PPP is the IETF Point-to-Point Protocol. [5,6] PPP-Networking is the means of taking IP packets to/from the PPP layer and placing them onto the LAN.

PPP is used to carry packets from the higher IP and WAP layers across Bluetooth's RFCOMM serial port emulation layer. In other words, to use IP with Bluetooth we must use a compatible protocol like PPP. RFC1661 describes PPP, and RFC 1662 describes how it can be carried across High level Data Link Control (HDLC) framing systems. (RFCOMM is an HDLC framing system).
PPP encapsulates information for transmission in RFCOMM frames as shown in figure below

Protocol Field(8 or 16 bits)	Information (0 or more bits)	Padding (optional)

Table 9: The PPP protocol format

The protocol field is used to identify what sort of PPP message is being sent. The categories are: Padding protocol, Link Control Protocol (LCP), password authentication protocol (PAP), link quality report and challenge handshake authentication protocol (CMAP).

To set up a PPP link first, an RFCOMM connection is established. Then each side sends LCP packets for link test and configuration. Different parameters can then be configured like Maximum transfer unit (MTU), authentication and address and control field compression for the data link layer
After configuration, packets can then be sent across the link, which configures the higher layer protocols like IP.

To summaries some reasons why PPP is a reasonable choice to use, we have the following:

- It uses some kind of autoconfiguration
- It implies security aspects like authentication and access control and encryption
- It is easy to deploy
- It has multi-protocol facilities and supports IP, IPX etc
- It is widely accepted, used and supported by vendors.

Proposal of how to run IP over PPP

In the first place, it is most natural to believe that a solution with IP over PPP will be solved straight forward like this kind of topology (see figure below). The Access Point includes a PPP server and IP will transport packets over to the Ethernet.

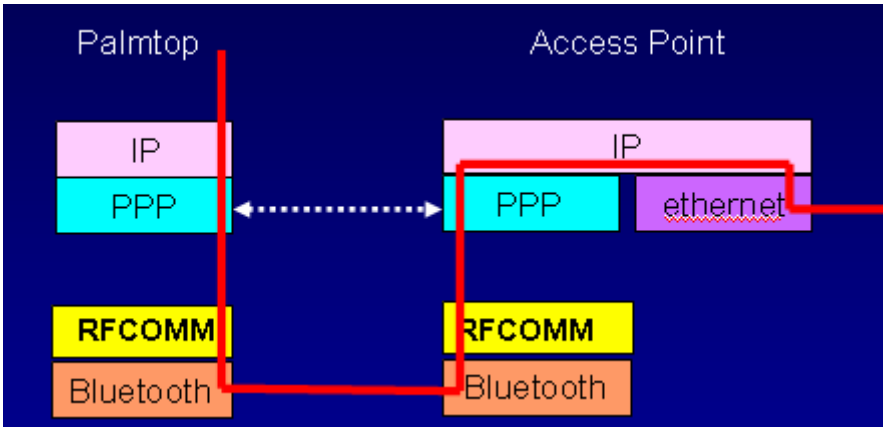


Figure 28: The usual way to run IP over PPP

This simple solution is fair enough but there are some problems.
To realize this, we have to have a PPP server function in every access point, which is not very favorable.
Another issue is the management of name/password and roaming will not be seamless. We will have to terminate the PPP connection every time we switch an access point.
Some of the problems can be solved by using tunneling as we tried to indicate below.

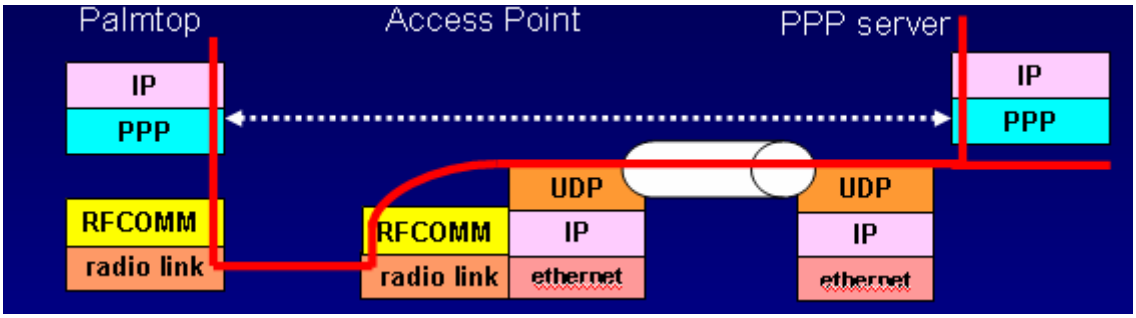


Figure 29: A better way to run IP over Bluetooth to solve the PPP problem

Every packet on a RFCOMM level is not unpacked any further at the access point. They are on the other hand forwarded in a tunnel through a wired network. A tunnel is an IP packet inside an IP-packet.

IP header for the Ethernet packet	Original IP packet from the Palmtop	IP tail for Ethernet packet

Table 10: a tunneled IP packet

If we use tunneling to tunnel all PPP traffic from the LAP to a PPP server we have a better solution.

- centralized management of user name/password
- reduction of processing and state maintenance at each access point
- seamless roaming

If one user suddenly switched an access point, the point to point connection will be terminated in the first solution. But since the PPP server is centralized we can maintain the PPP connection to the same server. Usernames and passwords can be centralized as well.
The first access point will send UDP packets to the PPP server automatic for every device with the original IP inside. When the second access point takes over, it will do the same thing. It will just forward the RFCOMM packets without reading them.
On the IP level, every packet will apparently be only connected to one server all the time.

We can not see that this solution is widely implemented today. Most of the access points that exist at the moment are based on IP over PPP but not the tunneled solution. Why haven't the tunneled based method (that is better) been accepted in the marked? Is it because as users want a "one pack" solution, an access point with every thing in it?
There are some weaknesses thou that could be improved.

- PPP is not sufficient for ad-hoc networks that contain multiple wireless hops.
  We have restrict the network access point to be a master and everybody else to by slaves
- The PPP server provide address assignment but it's not the best solution for ad hoc networking
  There is no way we can run ad hoc networking without PPP servers involved. There is no reason to believe that every Bluetooth device will be implemented with a PPP server. Hence, IP networking is impossible in ad hoc networks.
- PPP standard architecture provides one process per client. But RFCOMM can multiplex several PPP connections and create different treads.
  In other words we have to make a multithread implementation of the stack to support several IP connecting.
- Inefficiency of layering
  It is very inefficient to have a packet, split it up in bytes and put it back together again.

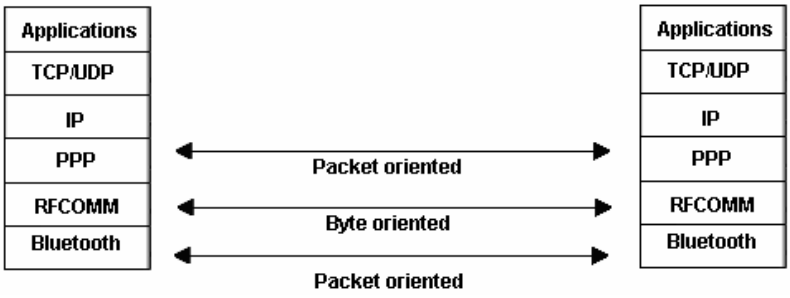


Figure 30: Inefficient layering

Relations with the Internet Protocol (IP)

Traditional IP is designed for stationary computers, and can't deal with the mobility demands of Bluetooth units. Issues like address resolution, bridging, multicast/broadcast mapping have to be solved. New protocols based on Mobile IP and Cellular IP are being studied, but not yet specified by SIG or IETF.
There is also a problem for TCP in wireless environments. Packet loss in TCP implies congestion in the TCP protocol. Usually TCP interprets packets loss as the result of congestion of the network and cuts back its transmit window size. This is true for wired networks, where the major cause of packets loss is buffer overflow. But for wireless links, this algorithm is not as effective, since most packet loss is caused by interference. Some new scheduling algorithms are being studied like other TCP variants.
There are a lot of IP issues that is important when we are designing IP over Bluetooth. We will now give a short evaluation of how different IP related protocols are affected by running IP over PPP over RFCOMM. We will be discussing subjects like DNS, DHCP, Address resolution, multicasting etc.

DHCP

The IPCP Protocol

The Point-to-Point Protocol (PPP) provides a standard method of encapsulating Network Layer protocol information over point-to-point links. PPP also defines an extensible Link Control Protocol, and proposes a family of Network Control Protocols (NCPs) for establishing and configuring different network-layer protocols.
The IP Control Protocol is almost the same as the Link Control Protocol.

IPCP Connection

IPCP (IP Control Protocol) only starts to operate after (a) the PPP connection has been established using LCP (Link Control Protocol) and optionally (b) the user has been authenticated. The IPCP protocol negotiates certain parameters between the LAP and the DT. Once the IPCP connection is established, and a suitable IP address has been negotiated, then the IP interface is enabled.

IP Address Allocation

The DT will require an IP address in order to operate on the LAN. Current PPP implementations allow only three possibilities:

- The IPCP option is used to specify a pre-configured IP address. If this IP address is not suitable on the LAN, then the IPCP link will not be established.
- The IPCP option is used to request a suitable IP configuration from a PPP Server.
- The IPCP Mobile-IP options are used to request a specified IP configuration. When moving between access points on the same LAN, it may be advantageous for the DT to continue using the same IP configuration. This is a very useful option if want to use Mobile IP.

In other words we don’t need any special kind of DHCP server. The PPP connection will take care of the IP address assignment.

DNS and NBNS addresses

Optionally, the LAP support could include the IPCP extensions defined in RFC1877 (defined by Microsoft). These extensions define the negotiation of primary and secondary Domain Name System (DNS) and NetBIOS Name Server (NBNS) addresses.

NetBIOS over IP
The NetBIOS protocol is used by Microsoft Windows to implement many of its networking features. The NetBIOS protocol can be carried over IP packets as defined in the NetBIOS specification.

DNS over IP
The DNS protocol is a part of the session layer in the OSI standard and is laid over IP in the protocol stack. As long as the layers below DNS work, DNS will work as usual.

Broadcast, Multicast and anycast

It’s a self-contradiction to try to do a broadcast on top of a PPP-connection.
PPP is just a tunnel on top of another protocol in principle and does not support any kind of broadcast.
If the broadcast should work properly every access point should be a router. This is an expensive solution, but this is the way it have do be done. This disputes with the centralized PPP server.
It may be possible to implement a router in every access point and let the access point find out where to do broadcast or multicast. But in principle the PPP protocol is not made to support broadcast.

Roaming

Two major current limitations of Bluetooth are: that communication between devices must be direct and secondly that they do not support the movement of an active terminal from one network interface device to another. We will give an introduction to how the basic roaming mechanism is in Bluetooth and some problems around it. In proposal chapter we give an outline of how to solve the problems.

The basic mechanism of roaming

The basic mechanism doesn't assume much and is quite simple. The Bluetooth device is connected to an Access Point, and moves out of range. After some time, the Bluetooth stack in the device declares the link to be dead and closes the LMP connection. At this point, the L2CAP connection used by the network traffic receives an event (LP_DisconnectInd) and closes down. The higher layers are kept in suspended mode until the roaming procedure is completed (TCP/IP may timeout if we take too much time).

At this point, the Bluetooth device performs an Inquiry. If the inquiry doesn't find any Access Point, the node continues with Inquiry, up to the point where a timeout occurs and closes down the higher layer, the PPP.

If the Bluetooth device finds an Access Point, it connects to it and queries its SDP service record. If the "Service Name" of this Access Point is the same has the one of the previous Access Point, the device connects to it. If the "Service Name" is different, and if the device discovered other Access Points in the Inquiry process, the device should try to connect to those Access Points. In other words, we are using the "Service Name" as a logical network identifier. If the Device can't find an Access Point with the same "Service Name", it can either try to connect to one of the other Access Point, continue doing Inquiry, or return a failure to the user. An Access Point with a different "Service Name" is managed by a different authority and may not offer the same connectivity options, so there is no guarantee that we can connect to it, have similar services and resume its suspended connections.

When the device connects to the Access Point, it registers to it and opens a L2CAP connection with this Access Point, and then reconnects the suspended higher layer on top of this L2CAP connection. For PPP, we need to establish a new PPP connection (LCP and IPCP) and redirect IP traffic on top of this new PPP instance. This problem will we be solved in another way of carrying IP over Bluetooth [7]. For the Access Point, it is as if the device would connect to it for the first time, and it just needs to pass the IP traffic back and forth. The old Access Point would also timeout, close the L2CAP connection and the higher layers, and flush its data buffers.

Discovery latency - Usually, a discovery is composed of 3 phases, Inquiry, Paging and higher layer connection. The Inquiry is usually the slowest and a full Inquiry as defined in the spec last in excess of 10 s. Such a long time has a high probability to interfere with the user experience and TCP/IP connections.

The remaining part of the Discovery is faster. Paging, when done after a discovery, is pretty quick (below 100 ms). Connecting the higher layer is pretty quick as well, establishing a L2CAP depends on how many L2CAP options are exchanged but should be well below 50 ms, for PPP it should be below 100 ms.

A normal procedure going from Start of Paging to IP flowing should be able to go in less than 250 ms. This kind won't affect the TCP/IP too much.

The discovery latency is mostly due to the Inquiry process.

When to do handover – Normally in mobile systems the RSSI (Received Signal Strength Indicator) value is used to decide when to do handover and switch Access Point. But Bluetooth devices don't have this value. This will be discussed in our proposals chapter in the end.

Routing

Routing in a Bluetooth Network, like any other type of Master-Slave network, is accomplished by a single device acting as the Master. This is the reason why a network access point must always be the master. The Master must maintain a table of IP addresses to link-layer addresses for every attached IP capable device. The mapping may be an IP address to a logical link-layer address (link local), or a physical link-layer address. An implementation may use a logical link-layer address such as the logical Connection Identifier (CID) provided by L2CAP.

For a dynamically assigned IP address:

IP address	CID	Bluetooth address

For a permanently assigned IP address, the IP address may be used to Uniquely identify a device:

IP address	CID

The problems arise when several Piconets interconnects. There have been several solutions of which dynamic routing algorithm to use in Bluetooth, but none of them have yet been standardized. We will come back to some problems concerning this topic later in chapter 6 and 7.

To implement a router in every access point is very expensive and hopefully we want to eliminate the router and replace it with a bridge.
If we want to use a bridge new problems arise like address resolution. Common address resolution uses broadcast and broadcast is not suitable in this solution.

Summary

To run IP over PPP over RCOMM is today a well implemented solution and is a part of the LAN access profile. Today this method is well known in the marked and several LAP is developed upon this idea.
PPP is a widely used protocol and is really good for IP and other protocols, but the method does not support ad-hoc networks and PPP is server dependent. Besides this, it takes a while to connect and process through all the protocol layers. There is a lot of overhead and RFCOMM is in our opinion too complicated and has a lot of unnecessary things. In addition, this is not the ideal solution since L2CAP is packet switched and RFCOMM byte switched. There are also some problems concerning IP protocol issues like broadcast and the needs of routing.
We can catch a glimpse of a more effective way to solve this problem. The following chapters contains other ways to build the stack. Later we will also see how the PAN profile works together with the BNEP protocol.

4 Other solutions to run IP over Bluetooth

Objectives

- Some other proposals of how to run IP over Bluetooth
- Evaluation of IP over L2CAP

Contents

Proposals of a new Bluetooth stack

In the Bluetooth technology, The PPP (Point-to-Point Protocol) is designed to run over RFCOMM to accomplish point-to-point connections. The current IP over Bluetooth solution known as the LAN Profile is a legacy solution designed for backward compatibility based on the serial cable replacement protocol RFCOMM. The IETF PPP protocol is used exclusively to transport IP traffic to and from the clients to the AP and out to the wired network. While this design is basically effective, all the limiting problems of running a network over serial PPP link are evident. Moreover PPP is not sufficient for ad-hoc networks that contain multiple wireless hops and other things we discussed in the previous chapter.
However, the Bluetooth specification is open (although SIG hasn't released all their work yet) and it is also possible to configure IP directly over L2CAP or other layers above the hardware (the HCI layer).
We have taken several possibilities into account but we have emphasized some solutions more than others.
Specially IETF and SIG have done a lot of research in this area, but also other research groups has given some proposals to a new and better Bluetooth stack.
Some of the methods are not very well documented and difficult to understand, while others are good.
The following list contains some of the proposals we have analysed.

Solution for IP under development by Bluetooth SIG

SIG [1] is at the moment working on a new profile. Their profile is not yet completely released and has still some work left.

SIG solution 1

The solution under development by the Bluetooth SIG is known as the PAN profile. We consider this solution as future oriented and will come back to this solution later.

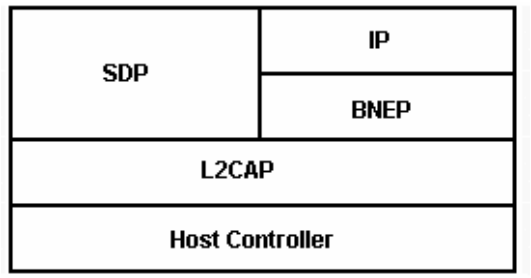


Figure 31: Proposed Bluetooth stack for IP networking by SIG

Possible solutions for consideration by the IETF

The IETF has suggested different ways to run IP over Bluetooth. The problem with many of their methods is that they will invent some new protocols and they are not specified or documented. To give an evaluation of these methods is not so easy, but we can catch a glimpse of some good opportunities.

IETF solution 1

The IETF may consider to run IP over modified L2CAP with, or without compression
Notice the modified SDP protocol as a new Service Location Protocol.

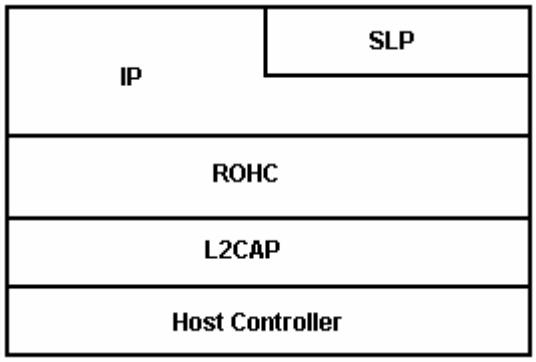


Figure 32: Proposal 1 for a possible IP stack

IETF solution 2

The IETF may consider running IP and L2CAP over the Host Controller with a new protocol that controls them both.

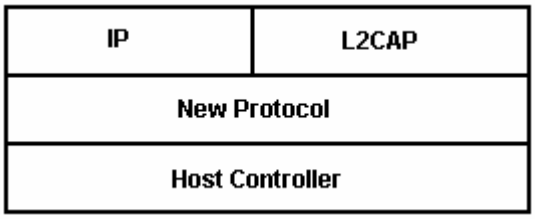


Figure 33: Proposal 2, possible IP stack

IETF solution 3

The IETF may consider running IP over L2CAP over HCI with a new protocol.

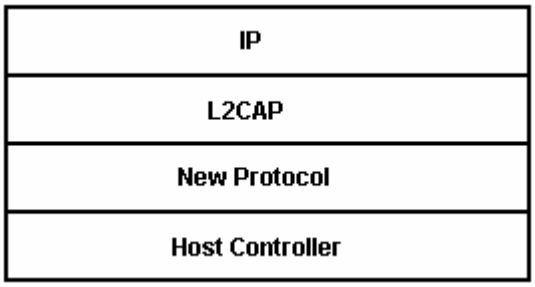


Figure 34: Proposal 3, possible IP stack

IETF solution 4

This solution is simple (and might be effective) and is a released draft from the IETF, May 2001. We will come back to this method later.

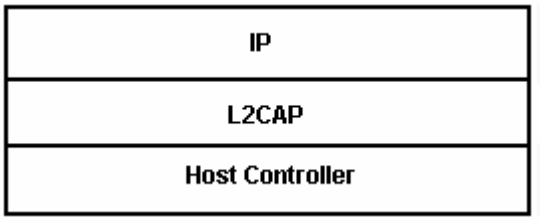


Figure 35: IP directly over L2CAP

Other proposals

PPP over L2CAP solution
Some people will go back to PPP again and use this solution. We find it preferable not use PPP to avoid ad-hoc networking problems, as we have discussed earlier.

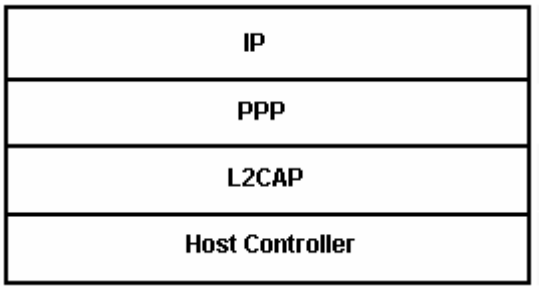


Figure 36: PPP over L2CAP

We have decided to take a closer look at 2 of these new methods, as these are best documented. We have already looked at the LAN Access Profile (IP over PPP over RFCOMM) and mentioned the Personal Area Network (PAN) profile.
In addition we find the IP over L2CAP method quite interesting.

IP running over PPP over RFCOMM, enables smooth operation and interoperability with legacy applications. In this case, the framing information available through HDLC (Higher Level Data Link Control) in PPP should be passed to the L2CAP layer to make it aware of PPP packet boundaries, thus enabling efficient SAR (segmentation and reassembly).
TCP/IP layered directly over L2CAP has lesser overheads, but overheads due to RFCOMM and PPP headers (14 bytes) are negligible compared to the size of the TCP packet and the delay involved.

It's possible to run TCP/IP over L2CAP directly, but currently, there are no profiles defined for them yet. So far, most vendors' TCP/IP implementations are based on PPP running on RFCOMM, but in the future we think there will be may be 3 or more ways to run IP over Bluetooth. LAN Access Profile, PAN Profile and may be IP over L2CAP.

The next chapter will give a short evaluation of how to run IP over L2CAP.

Running IP over L2CAP

This solution is based on a draft released from the IETF, May 2001. It treats Bluetooth as another link layer over which all IP compatible protocols are carried. Also, this approach allows an ad hoc network to be treated as an isolated IP capable network, rather than as a special case requiring special protocols as in the LAN Access profile.

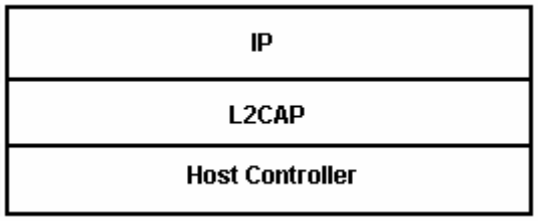


Figure 37: IP over L2CAP

The proposed solution will have the following stack on an IP Router attached to an IP capable wired or wireless network.

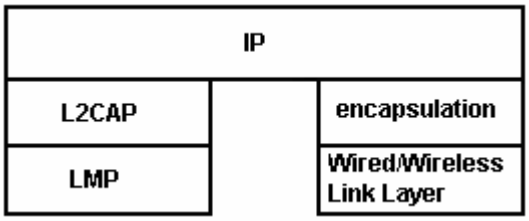


Figure 38: Router connection

The encapsulation protocol in Bluetooth, L2CAP, provides a Channel Identifier (CID) field for the dynamic encapsulation of other protocols. At this point we assume that IP is set up by the Service Discovery Protocol (SDP).

Length	Channel ID	Payload
16 bits	CIP-IP	IP packet

Figure 39: A L2CAP packet, encapsulating the Internet Protocol

The Maximum Transmission Unit for an L2CAP packet is 65535 bytes and an IP packet is 64KB. Therefore, at this point there is no segmentation problem.

The Bluetooth link-layer is connection-oriented and any two devices that wish to communicate must setup a point-to-point connection. One as Master and one as Slave. The master may have more than one point-to-point connection active at any time. A connection must be setup before any IP packets can flow.

IP ISSUES

Bluetooth addressing

If the Bluetooth link-layer address of the other device is not known previously, the Inquiry procedure must be preformed by the device requesting a connection. If the link-layer address is known, then the requesting device can just perform the Page procedure. In any case, after the connection setup process is complete both devices will know the link-layer address of the other. Therefore, a link-layer address resolution protocol like ARP is not necessary within a Piconet (and only within a piconet). The Master holds all link-layer addresses within the Piconet as a result of connection setup.
But if the network is larger, like a scatternet, and two devices in two different interconnected piconets want to connect each other, we need some kind of ARP mechanism to get access to the link layer address from the other piconet.
The ARP packet will be encapsulated in a L2CAP packet, and broadcasted over the Bluetooth network via Bluetooth broadcast.
L2CAP provides a Protocol Service/ Multiplexer (PSM) field for encapsulation of the Address Resolution Protocol. The PSM field can be used for the encapsulation of the Address Resolution.

Length	Channel ID	Protocol Service	Payload
16 bits	CIP-IP	PSM_ARP	ARP packet

Figure 40: A Bluetooth ARP Packet

The problem is that this is a special Bluetooth ARP packet called BARP. ARP only works within a Bluetooth network. (Just as an LAN ARP packet only works within the LAN). Hence, we must still have routing capability in the access points.

Routing

Routing in a Bluetooth Network, like any other type of Master-Slave network, is accomplished by a single device acting as the Master. The Master must maintain a table of IP addresses to link-layer addresses for every attached IP capable device.

Broadcast

The slots must be used for a Piconet Active Member Broadcast. If some of the participants are “Parked” devices, these participants should be made active, and given sufficient bandwidth for the Broadcast. If it is not possible to “un-Park” all the participants, then a Piconet Broadcast shall be used.

Summary

When we use this method, we do get a more efficient Bluetooth network, with less overhead. The method has not yet been specified, and we think it never will. The IP over L2CAP method still requires a lot of processing in the Access Points. The goal is to make every access points more like a bridge instead of a pure router.
The master-slave topology is a problem in an IP network. The goal is to eliminate the master-slave structure and let every node act like peers. IP directly over L2CAP will not solve this problem.
On the other hand, BNEP is assumed to gratify these conditions. It will even let LAN IP nodes act as they are one the same network.
The new PAN profile (BNEP) is building on the “IP over L2CAP” method. The work on this solution decreased and at the moment the focus is in the new and better solution by means of BNEP.

5 The PAN profile and the BNEP protocol

Objectives

This chapter describes the Pan Profile and how to use the Bluetooth Network Encapsulation Protocol (BNEP) Specification to provide networking capabilities for Bluetooth devices. We will address the following

- The Pan Profile
- Ethernet Encapsulation(BNEP)
- Ad-hoc networks versus Network Access Point, bridging/routing
- Discussion of different IP issues.

Contents

The PAN profile

The PAN profile lays out the rules for carrying IP traffic across Bluetooth connections. This is done using Ethernet like packets encapsulated in L2CAP packet payloads using the Bluetooth Network Encapsulation Protocol (BNEP). The PAN profile relies upon the Generic Access Profile, but not makes use of any other profiles. The profile provides two ways of connecting. Devices can connect to a Network Access Point (NAP) in order to access remote networks. The Network Access Point provides bridging capabilities to outside networks. Alternatively, devices can connect directly to one another using Group ad-hoc Networks (GN)

In general we will discuss two types of scenarios:

- Network access points
- Group Ad-hoc Networks.

Network Access Points

In this scenario, the radio and host controller appear to be a direct bus connection to a network interface device with network access. A network access point is a device that contains one or more Bluetooth radio devices and acts as a bridge, proxy, or a router between a network (10baseT, GSM, etc) and the Bluetooth network. Each network access point can allow one or more computing devices to gain access to it, and each of these computing devices will have access to all of the LAN's shared resources. Network access points can provide access to other networks technologies such as ISDN, GPRS, Home PNA, and Cable Modems and cell phones.

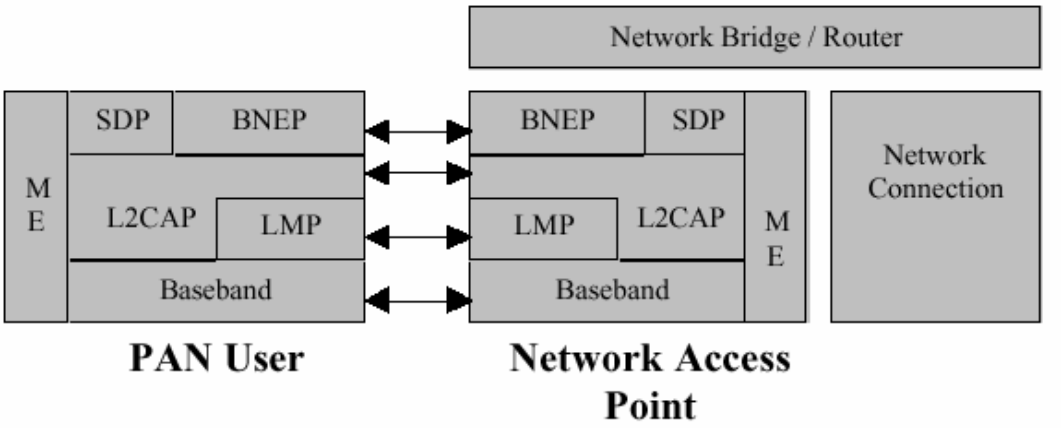


Figure 41: NAP Bridge

Note that ME is a management entity that controls procedures during initialization, configuration etc.

Group Ad-hoc Networks

Group ad-hoc networking is a collection of mobile hosts that cooperatively create an ad-hoc wireless network without the use of additional networking hardware or infrastructure. In addition, the PAN profile focuses on the following simple personal ad-hoc networking scenarios consisting of a single Bluetooth piconet, with connections between two or more Bluetooth devices.
A piconet consists of one Bluetooth device operating as a piconet master communicating with between 1 and 7 active Bluetooth devices operating as slaves. Communications in a piconet are between the master and the slaves and under the control of the master either in a point-to-point or point-to-multi-point fashion. In addition, there may be further non-active piconet members that are in park mode. The limitation of 7 active slaves in a piconet is enforced by the Bluetooth active member-addressing scheme. A group ad-hoc network is a set of computing devices which interact with each other to form a self-contained network, the network being formed without the need for additional external networking hardware.

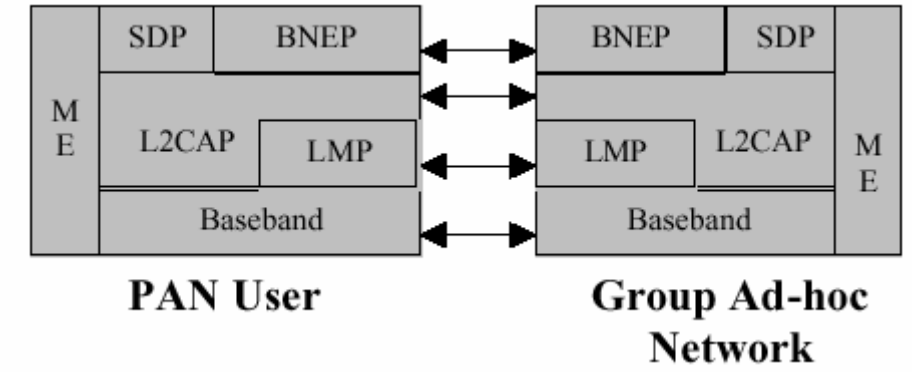


Figure 42: Group Ad-hoc networking

Configurations and roles for a Bluetooth device

These scenarios define 3 types of roles a for a Bluetooth device:

1. a Bluetooth device communicating with a NAP (network access point)
2. a Bluetooth device communicating with other Bluetooth devices (group ad-hoc network)
3. a Bluetooth device communicating with both NAP and other Bluetooth devices

Role 1: BN – NAP

- The first step is to find a NAP that is within radio range and provides the NAP service. For example, the PANU could use an application to inquire for nearby devices and then use SDP to retrieve records that support the NAP service.
- If there is no existing Bluetooth connection, then the PANU requests a Bluetooth connection with the selected NAP. If the NAP is in multi-user mode, an M/S switch will be required to complete the connection.
- Once the connection is made, the PANU (Pan User) shall create an L2CAP channel for BNEP and may use the BNEP control commands to initialize the BNEP connection and setup filtering of different network packet types. The NAP would store all network packet type filters in a Packet Filter Database, which would maintain a set of packet filters for each connection.
- Ethernet traffic can now flow across the link. The PANU uses the services provided by the remote network, such as obtaining an IP address by using DHCP (Dynamic Host Configuration Protocol) for example. Other network services of interest can also be used by the PANU. The NAP will forward the appropriate Ethernet packets to each of the connected PANUs and over the NAP network connection. This behavior is similar to a network hub.
- At any time the PANU or the NAP may terminate the connection(s).

Role 2: BN – BN

- The first step is to find another Bluetooth device that is within radio range and provides the GN service by using baseband inquiries and SDP searches.
- If there is no existing Bluetooth connection, then the PANU requests a Bluetooth connection with the selected device providing the GN (general ad-hoc network) service. A master-slave switch will be required to complete the connection if the GN is in multi-user mode.
- Once the connection is made, the PANU can create an L2CAP channel for BNEP and use the BNEP control commands to initialize the BNEP connection and setup filtering of different network packet types. The GN would store all network packet type filters in a Packet Filter Database, which would maintain a set of packet filters for each connection.
- Ethernet traffic can now flow across the link. GNs will not provide networking services and therefore each of the PANUs will perform various tasks to operate without these services, for example Autonet . The GN will forward all Ethernet packets to each of the connected PANUs.
- At any time the PANU or the GN may terminate the connection(s).

Role 3: BN - BN – NAP

- The first step is to find another Bluetooth device that is within radio range and provides the PANU service by using baseband inquiries and SDP searches.
- If no Bluetooth connection exists, then the GN requests a Bluetooth connection with the selected device using the PANU service. No master-slave switch will be required.
- Once the connection is made, the GN can create an L2CAP channel for BNEP. The PANU uses the BNEP control commands to initialize the BNEP connection and setup filtering of different network packet types. The GN would store all network packet type filters in a Packet Filter Database, which would maintain a set of packet filters for each connection.
- Ethernet traffic can now flow across the link. GNs will not provide networking services and therefore each of the PANUs will perform various tasks to operate without these services, for example Autonet. The GN will forward all Ethernet packets to each of the connected PANUs.
- At any time the PANU or the GN may terminate the connection(s).

Bluetooth Network Encapsulation Protocol

The Bluetooth Network Encapsulation Protocol (BNEP)[7] encapsulates packets from various networking protocols, which are transported directly over the Bluetooth Logical Link Control and Adaptation Layer Protocol (L2CAP) protocol. BNEP is used to transport both control and data packet over Bluetooth to provide networking capabilities for Bluetooth devices. BNEP provides capabilities that are similar to capabilities provided by Ethernet. The IP layer above BNEP in turn supports a session management layer. This could be Transport Control Protocol (TCP) or User Datagram Protocol (UDP).

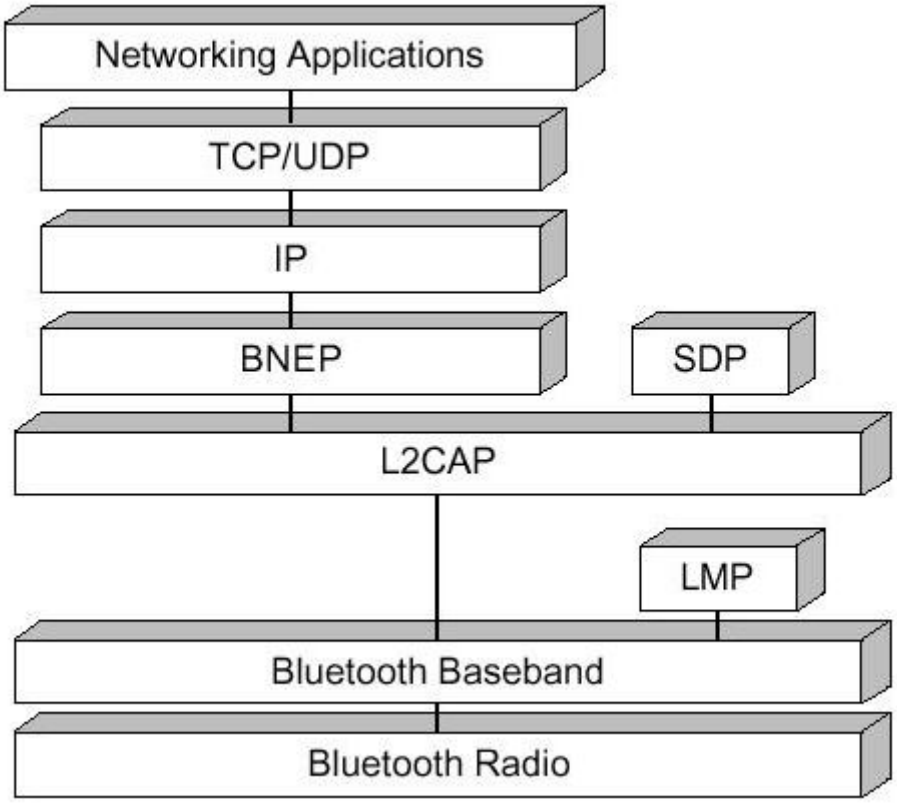


Figure 43: BNEP protocol stack

It is possible that the device terminating the BNEP link acts purely as a router, just forwarding IP packets onto an Ethernet. Packets can be forwarded by the IP layer, never reaching the TCP or UDP layer. Some IP router implementations omit the higher layers. These implementations have protocol stacks which stops at the IP layer and omit the session layer. In BNEP a session must be established and managed, as a L2CAP channel must be set up. So even if a full UDP or TCP implementation is not required some session management functions will be needed to manage the Bluetooth connections.

Packet Encapsulation

The use of the BNEP for transporting an Ethernet packet is shown in Figure 45. BNEP removes and replaces the Ethernet Header with the BNEP Header. The Ethernet Payload remains unchanged. Finally, both the BNEP header and the Ethernet Payload are encapsulated by L2CAP and sent over the Bluetooth media. The maximum payload that BNEP will accept from the higher layer is equal to the negotiated L2CAP MTU (minimum value: 1691 bytes), minus 191 bytes (reserved for BNEP headers). This way it can be assured that enough frame buffer space is reserved to transmit all BNEP.

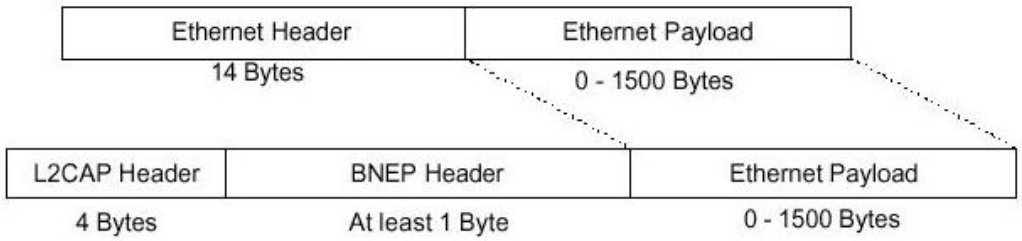


Figure 44: BNEP encapsulation

BNEP Header Formats

All BNEP Headers are in the following format as shown in Figure 46.

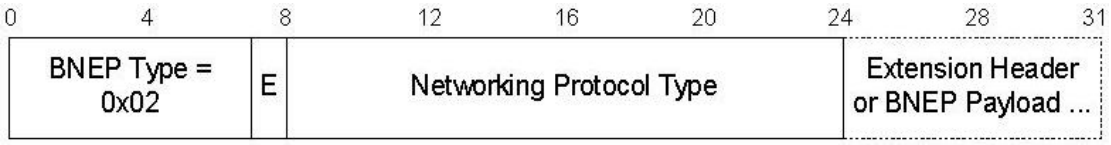


Figure 45: BNEP header

The seven bit Bluetooth Network Encapsulation Protocol Type value identifies the type of BNEP header contained in this packet.

Value	BNEP Packet Type
0x00	BNEP_GENERAL_ETHERNET: This packet type shall be used to carry Ethernet payloads to and from Bluetooth networks.
0x01	BNEP_CONTROL: This packet type is mandatory to recognize and respond to accordingly, this packet is used to exchange control information like connection setup and filter settings. An example is connection request and connection response messages.
0x02	BNEP_COMPRESSED_ETHERNET: This packet type shall be used to carry Ethernet payloads to and from devices that are directly connected at L2CAP level using BNEP.

0x03	BNEP_COMPRESSED_ETHERNET_SOURCE_ONLY: This packet type will be used to carry Ethernet payloads to a device using BNEP, which is the final destination for that packet. Devices do not need to include the destination address in the packet, because the destination address of the BNEP packet is the same as the address corresponding to the L2CAP channel over which the packet is sent.
0x04	BNEP_COMPRESSED_ETHERNET_DEST_ONLY: This packet type shall be used to carry Ethernet payloads from a device using BNEP, which is the originator of that packet. Devices do not need to include the source address in the packet, because the source address can be determined from the L2CAP connection and which device sent the packet.
0x05 – 0X7E	Reserved for future use
0x7F	Reserved for 802.2 LLC Packets for IEEE 802.15.1

Table 11: BNEP header formats

The BNEP_COMPRESSED_ETHERNET header format is based on one of the compressed versions of the Ethernet header supported by BNEP. This packet type shall be used to carry Ethernet packets to and from devices that are directly connected at L2CAP level (have a valid L2CAP connection) using BNEP. This compressed header may be used when two Bluetooth devices are exchanging packets, in which the source address is set to the local device's address which is the source device sending the packet and destination addresses are set to the other device's address which is the final destination for the packet. Devices do not need to include the source or destination addresses in the packet because the destination address is always the device's address that received the packet and the source address is always the device's address that sent the packet. All devices supporting BNEP must be able to interpret all defined BNEP packet types. We can se the use of all different packet types. BNEP general Ethernet packet payload is mandatory for sending an IP packet from an outside network to the Bluetooth Units PAN. All traffic must go through the PANs master. BNEP control messages are mandatory for setup and filtering. The BNEP COMPRESSED ETHERNET, COMPRESSED ETHERNET SOURCE ONLY, BNEP COMPRESSED ETHERNET DEST ONLY are not mandatory but decrease the BNEP overhead and transmission time.

Extension flag E:

Value	Parameter description
0x0 – 0x1	One bit extension flag that indicates if one or more extension headers follow the BNEP Header before the data payload. If the extension flag is equal to 0x1 then one or more extension headers follows the BNEP header. If the extension flag is equal to 0x0 then the BNEP payload follows the BNEP header.

Table 12: BNEP extension flag

BNEP overhead

- 7-bit Bluetooth value identifies the type of BNEP header contained in this packet
- 1-bit extension flag that indicates if one or more extension headers follow the BNEP Header before the data payload and is defined for possible future extension of the BNEP.
- Additional ~0.2% BNEP overhead (24 bits without extension header)
  -Additional Bluetooth Transmission time: 11 mSec (Bluetooth SIG)
- There is a lot of reserved packet types that may be used for future expansion
- The L2CAP packet length can be used to determine the total length of the BNEP packet. Therefore a length field in the BNEP header is redundant.

IP issues in BNEP

BNEP encapsulates IP packets to the correct network. With BNEP all Bluetooth Devices should become peers with outside networks. To make a Bluetooth device become a peer in an Ethernet segment (IEEE 802.3) BNEP must support all IP related protocols like IEEE802.3.
In this chapter we will evaluate different IP issues in two scenarios; Infrastructure mode and ad-hoc mode, but first some general issues about ARP.

ARP (Address Resolution Protocol)

The Address Resolution Protocol (ARP)[15] is a protocol for mapping an Internet Protocol address (IP address) to a physical machine address that is recognized in the local network. For example, in IP Version 4, the most common level of IP in use today, an address is 32 bits long. In an Ethernet local area network, however, addresses for attached devices are 48 bits long. (The physical machine address is also known as a Media Access Control or MAC address.) A table, usually called the ARP cache, is used to maintain a correlation between each MAC address and its corresponding IP address. ARP provides the protocol rules for making this correlation and providing address conversion in both directions.

ARP in BNEP

ARP support is mandatory for BNEP, Since Bluetooth Devices have Bluetooth addresses and not MAC addresses there should be a problem with an ARP request when mapping IP to the MAC address. BNEP solves this problem by using the Bluetooth address as MAC address, the Bluetooth address is 48 bits like an ordinary MAC address and it is unique. This makes ARP feasible in BNEP.

How BNEP works in an infrastructure scenario

In an infrastructure scenario, the Pan Users (PANU) are connected to a Network Access Point (NAP) which provides network services to all Bluetooth devices connected. The NAP work as a bridge to other networks outside the PAN. The NAP will remove the BNEP header and put on correct header.

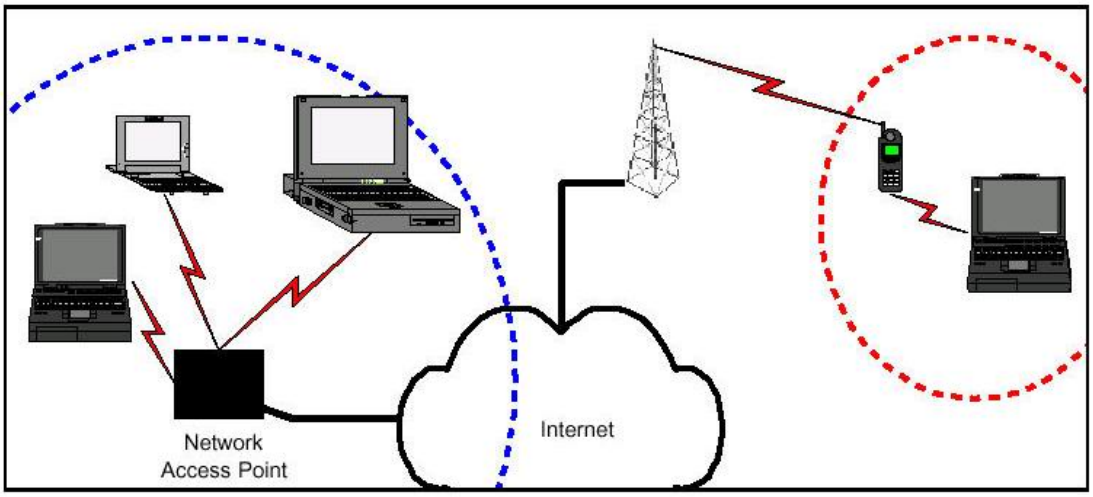


Figure 46: Infrastructure scenario

We will now show how different tasks are done by using BNEP in an infrastructure scenario[3]. In all scenarios the devices must be within radio range of the NAP, the NAP must be able to route/bridge traffic between the PANU and the respective server/network, and the network connectivity must be ensured between these two devices. The NAP and the PANU must maintain an open L2CAP connection.

IP address assignment (DHCP)

DHCPDISCOVER, DHCPOFFER, DHCPREQUEST and DHCPACK packets must conform to their respective specifications. The two functions (NAP and DHCP Server) may or may not be on physically separate devices. If they are located in different devices, network connectivity must be ensured between these two devices.

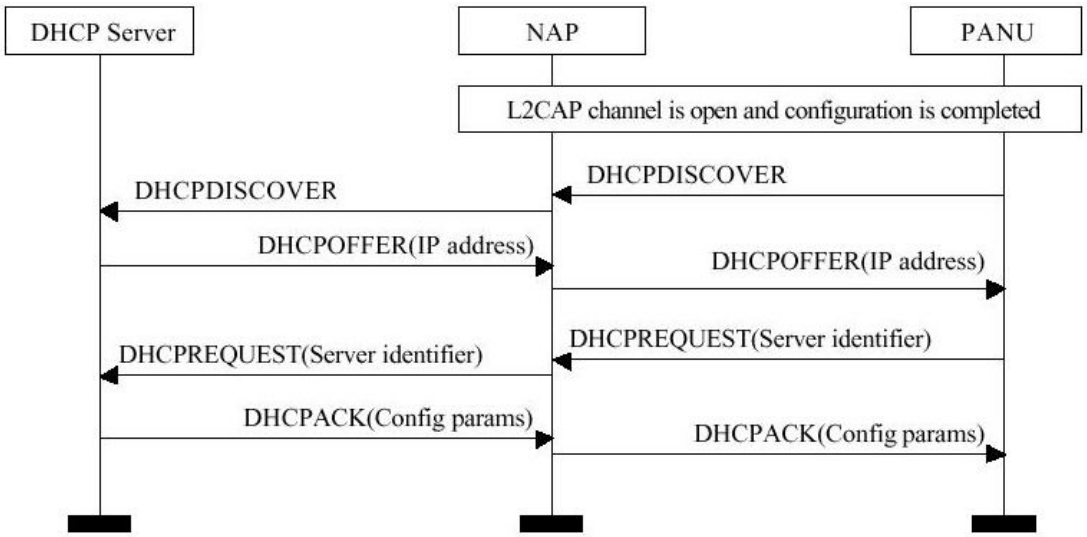


Figure 47: Infrastructure DHCP

1. PANU broadcasts a DHCPDISCOVER packet onto its local subnet.

2. The DHCP server transmits a DHCPOFFER packet containing an available IP address as well as additional configuration parameters to the PANU.

3. The PANU responds by broadcasting a DHCPREQUEST with an identifier for the selected DHCP server (to handle the case where multiple DHCP servers responded on the DHCPDISCOVER).

4. On receipt of the DHCPREQUEST the selected DHCP server responds with a DHCPACK with the configuration parameters for the PANU.

Broadcast from Bluetooth network to wired networks

Initial condition:

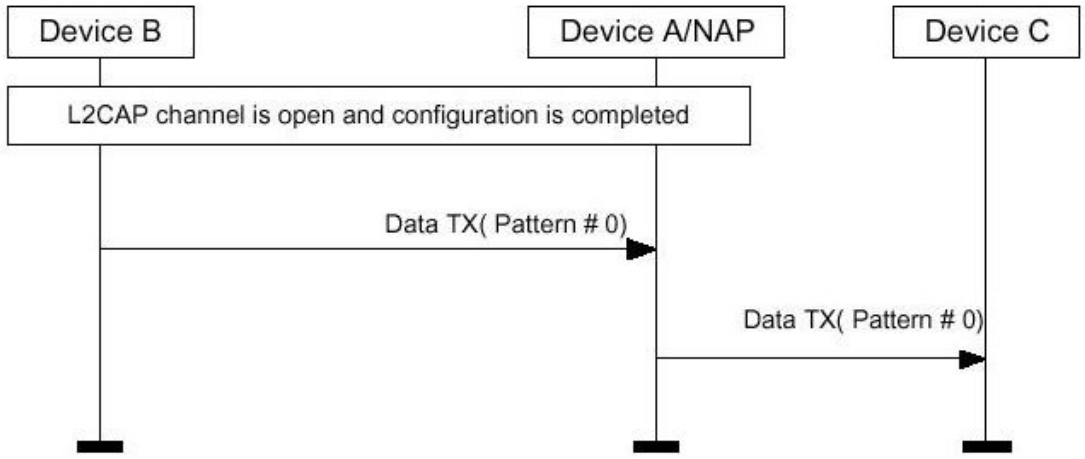


Figure 48: Infrastructure Broadcast

This involves a wired networked device (Device C) and two Bluetooth devices (Device A, B), Device C will act as the wired device connected directly to or via a hub to Device A. Device A will act as a NAP and remain as piconet master for the Bluetooth network, and connected via a wired network to Device C. Devices B will act as a slave. At first, Device B will send a BNEP packet (consisting of Broadcast IP packet) to Device A, the NAP. Device A will upon receiving the BNEP packet (broadcast packet) will remove the BNEP packet header and form an Ethernet packet to broadcast in the wired network. Device C on the wired network will receive the Ethernet broadcast packet.

Device B (Slave Bluetooth device) transmits a BNEP packet to Device A (NAP). Device A will remove the BNEP header, add Ethernet header and broadcast the packet to the wired network. Device C will receive the broadcast packet.

BNEP in ad-hoc networking

A piconet consists of one Bluetooth device operating as a piconet master communicating with between 1 and 7 active Bluetooth devices operating as slaves like figure 51

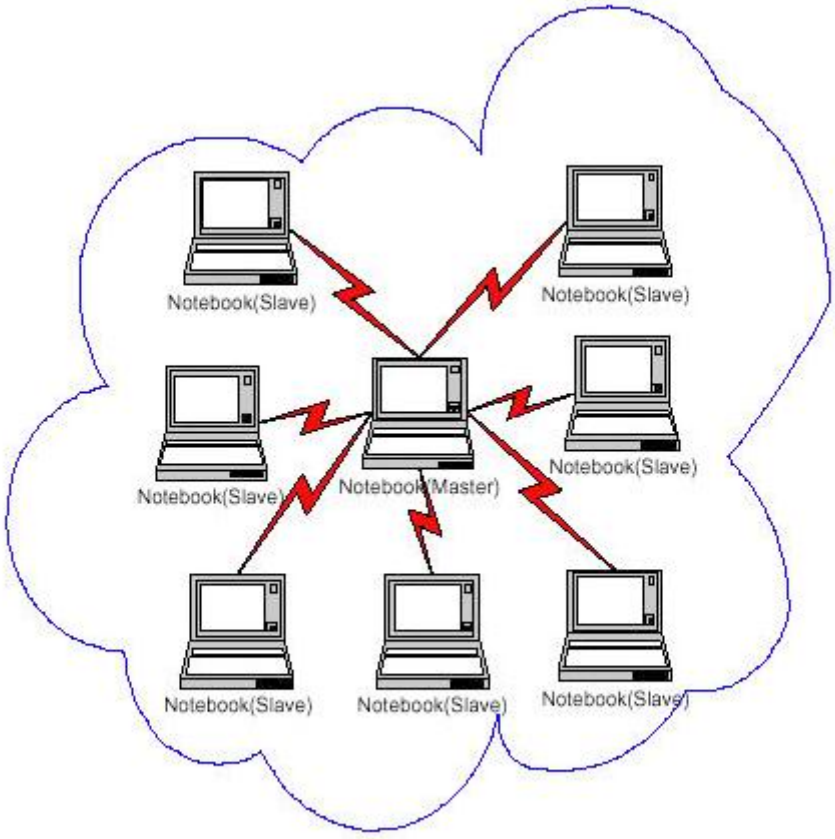


Figure 49: BNEP ad-hoc scenario

We will now show how different tasks are being done in an ad-hoc scenario[3].

Address assignment ad-hoc networking

Initial condition:

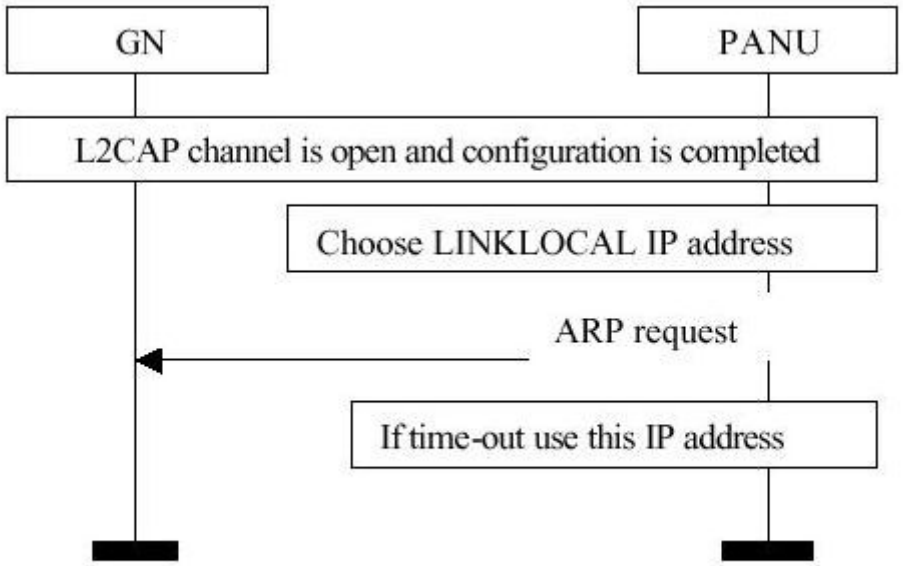


Figure 50: Ad-hoc address assignment

This will show how a PANU can obtain a valid IP address in an ad-hoc networking topology.

1. PANU chooses a valid LINKLOCAL IP address.

2. PANU must check if the selected IP address is already in use by means of issuing an ARP probe. If the selected IP address is already in use (or it receives another ARP probe with the same IP address in) it must select another IP address.

3. PANU must try finding a valid IP address until it either finds one, or it has tried more than autoconfig-retry count (default number, which tells how many times PANU should try to find a valid IP address). In the case where PANU acquires a valid IP address it must periodically try to fetch an IP address from a DHCP server, by transmitting DHCPDISCOVER packets.

Bridging in ad-hoc networking

Forwarding in general ad-hoc networking

This involves three Bluetooth devices that form one piconet. Thus two slaves are connected to the third device that is the piconet master. One of the devices in the role of slave sends a BNEP Packet to another Bluetooth device in the role of slave. The piconet master receives the packet from the originator and forwards it to the destination node.

Initial condition:

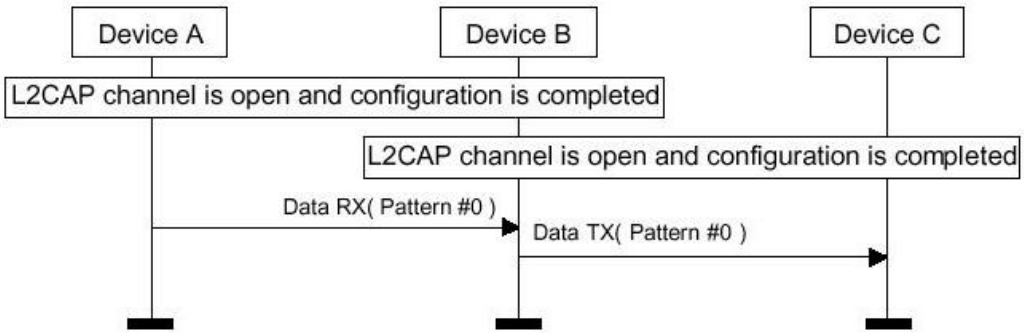


Figure 51: Forwarding in ad-hoc networking

Device B receives a packet from Device A, and forwards this packet to Device C. Device B is successful if Device B receives and interprets the BNEP header correctly and subsequently forwards it to Device C on the correct L2CAP channel.

Forwarding with BNEP compressed Ethernet

This packet type shall be used to carry Ethernet packets to and from devices that are directly connected at L2CAP level (have a valid L2CAP connection) using BNEP. This compressed header may be used when two Bluetooth devices are exchanging packets, in which the source address is set to the local device's address which the source device is sending the packet and the destination address is set to the other device's address, which is the final destination for the packet. Devices do not need to include the source or destination addresses in the packet because the destination address is always the device's address that received the packet and the source address is always the device's address that sent the packet.

Forward – Broadcast ad-hoc

This involves four Bluetooth devices that form one piconet. Thus three slaves are connected to the fourth device that is the piconet master. One of the devices in the role of slave sends a BNEP Broadcast Packet to all other Bluetooth device in the piconet. The piconet master receives the packet from the originator and forwards it to its other slaves. It also passes up a copy of the packet to its IP layer. This will show the correct use of the BNEP_GENERAL_ETHERNET (0x00) packet type for forwarding Ethernet broadcast packets in a Bluetooth piconet. The piconet master is the target device.

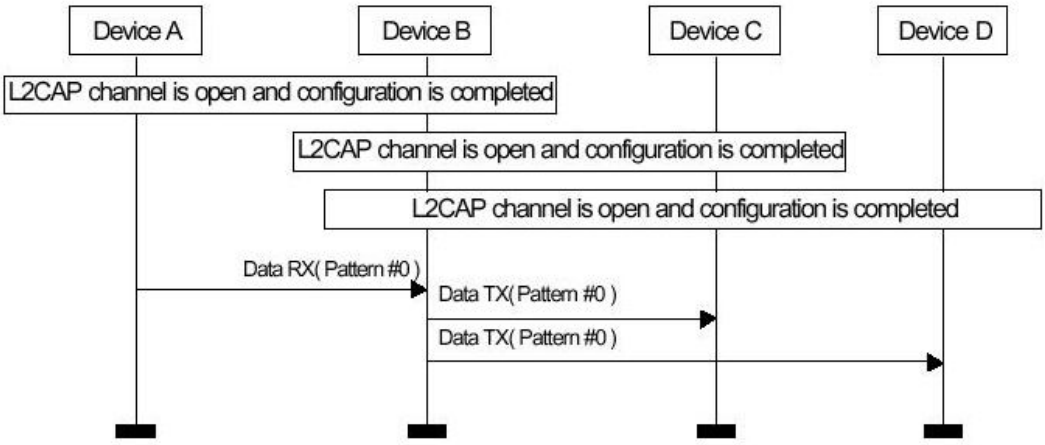


Figure 52: Forward - Broadcast ad-hoc

Device B receives a packet from Device A, and forwards this packet to Device C and Device D, as well as to its own IP layer.
Device B is successful if Device B receives and interprets the BNEP header correctly and subsequently forwards it to Device C and Device D on the right L2CAP channels. Device B must also remove the BNEP packet info from the data and pass a valid IP packet to the IP layer.

BNEP Summary

One of BNEPs intensions is to remove the master slave roles and make the Bluetooth devices becoming peers. BNEP adds an unavoidable additional encapsulation layer between the L2CAP encapsulation layer and the encapsulation layer provided by IP, such that there are three encapsulation layers, but only adds approximately 0.2 % overhead, and supports header compression.
BNEP supports IPv4, IPv6, IPX and other Networking protocols and all IETF protocols will work over Bluetooth without other changes. BNEP removes the Master Slave roles and makes all devices appear to be peers in an Ethernet segment (IEEE 802.3). With the NAP working as a bridge Bluetooth Devices can use all IP related protocols like ARP, DNS and DHCP. This makes BNEP to be probably the best currently solution for implementing IP over Bluetooth, BNEP are supposed to support different systems like GPRS and 3G Cellular in the future simply by removing the BNEP header and add the correct header. Unlike PPP, BNEP will support Broadcast and ad-hoc networking.
BNEP's low overhead and Ethernet's plug-and-play features match perfectly the characteristics of Bluetooth protocols such as baseband, link management, and service discovery. This presents a radical change from previous solutions in which Bluetooth was indeed treated as a "cable replacement" technology, leading to the continued use of heavy serial line emulation protocols on top of this new type of "cable".
BNEP is currently only defined for Piconets, and does not support Scatternets. The Scatternet solution is not a good solution though, we will discuss Scatternets in the next chapter, and make our own proposal with the use of BNEP to build a larger Bluetooth network.

6 Scatternet Issues

Objectives

- Problem because there is no support for IP scatternet yet.
- Present how scatternet works
- What are the problems with scatternet
- Routing versus forwarding in scatternet.
- Motivation not to use scatternet at all and rather only piconets.

Contents

Introduction

The way the BNEP is specified until this point, it has no support for Scatternets. The BNEP protocol only supports IP traffic within one piconet and IP bridging through a Network Access Point. There is currently a lot of work going on at SIG that deals with this problem.
This chapter will give a short introduction to how scatternet works, some of its approaches to the problem and why it is so. In this chapter we will discuss different problems and issues concerning scatternets.

Master/slave topology.

When a device is present in more than one piconet, it must time-share, spending a few slots on one piconet and a few slots on the other.

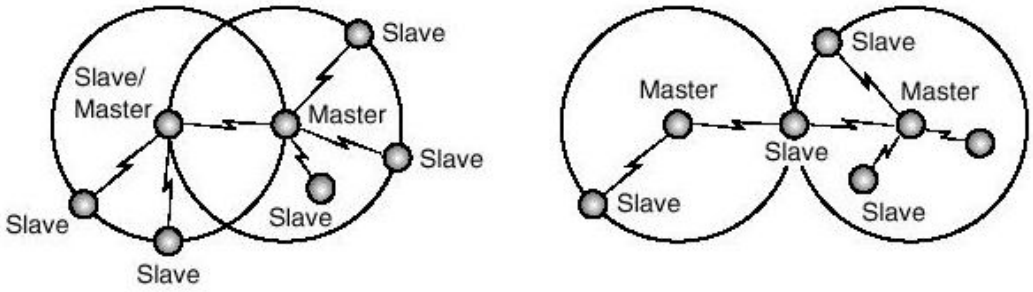


Figure 53: How Piconets can interconnect

On the left in figure is a Scatternet where one device is a Slave in one piconet and a Master in another. On the right is a scatternet where one device is a Slave in two piconets. It is not possible to have a device which is a Master of two different piconets and by definition, all devices with the same Master must be on the same piconet.
A major source of interference for Bluetooth devices will clearly be other Bluetooth devices. Although devices sharing a piconet will be synchronized to avoid each other, other unsynchronized piconets in the area will randomly collide on the same frequency. If there is a collision on a particular channel, those packets will be lost and subsequently retransmitted, or if voice, ignored. So, the more piconets in an area, the more retransmissions will be needed, causing data rates to fall. This is like having a conversation in a noisy room: the more people talking, the noisier it gets, and you have to start repeating yourself to get the point across.
This effect will happen if there are many independent piconets in one area, and it will also happen to Scatternets, since the piconets making up the scatternet do not coordinate their frequency hopping.

Master/slave switching and scheduling

A scatternet comprises two (or theoretically more) piconets, where at least one device is active in both. Switching between the piconets requires keeping track of two time bases, as each piconet is synchronized to a different Master's CLKN. A device which is active in two piconets must maintain and select between two piconets clocks, CLK1 and CLK2, and two access codes, CAC1 and CAC2.
The switch in time bases will require a guard time to synchronize to the new time base, as the Piconet's timings will be asynchronous to each other. In the worst case, they could be a whole slot pair out of sync. This puts a severe limit on how many piconets can be linked together if any meaningful degree of traffic is to be supported on each. SCO makes this especially difficult, as it uses reserved slots. Indeed, only one SCO link using HV3 packets may be active; even then, this only allows one other piconet to be visited in the four slots between the HV3 packets scheduled every six slots. Of the four intervening slots, two are used as guard slots and two are available for traffic in the other piconet.
Because of the necessity for guard slots created by this difference in slot timings, a device which is present on two piconets will never provide the maximum data bandwidth it would have were it only on one piconet.

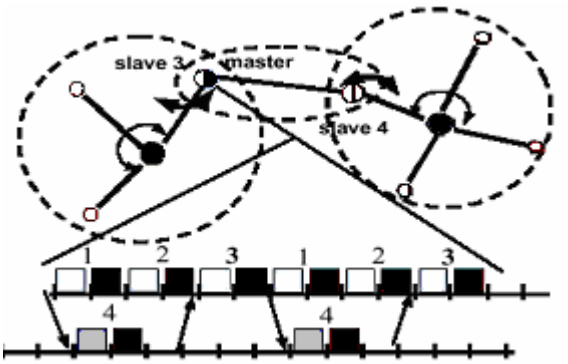


Figure 54: Interpiconet scheduling

When a device is busy on another piconet, it can no longer be contacted by its master and must therefore warn the master of the periods when it will be away, so that the master doesn't think it has lost communication with the Slave altogether. This may be done if the Slave negotiates entry into a low power mode with the master. In low power modes a device is allowed to switch off for some of the time. For a device which will regularly communicate on two piconets, sniff mode is probably the most useful since it allows a device to reduce communication to a regular agreed periodic slot sequence. Of course, in this case, the device is not going to sleep, it is in fact busy talking to someone else.
To summarize, the main problem is that the different piconets are using different Frequency Hopping Sequences (FHS) so the piconets are not synchronized. Due to this, we must introduce a guard time and we loose bandwidth

Sharing bandwidth.

A major technical step is taken when the Bluetooth piconet network architecture, a strict star topology, is extended into a Scatternet architecture, in which piconets are allowed to interconnect. The combinations of network topologies will increase dramatically and methods to create robust, but still efficient, Scatternets become crucial. To create good scatternet connectivity, bandwidth and delay requirements, among others, must be considered.

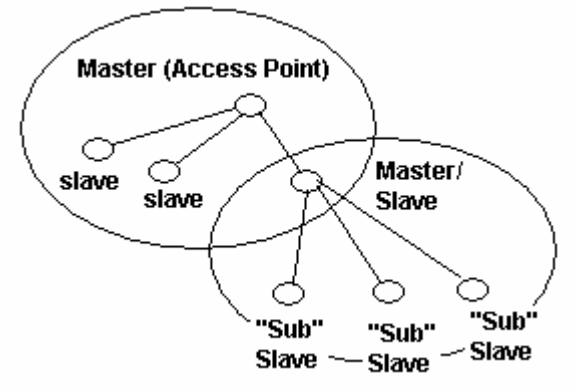


Figure 55: Bandwidth sharing within a scatternet

The figure shows that every what we call “sub” slaves must share the bandwidth and they only get 1/3 of the bandwidth of what the other slaves have. The problem gets even worse if the “sub” slave it self also becomes a master in another piconet. The scheduling will not be fear, but in some instances that may be desirable?

Low-power devices

In many cases, the network nodes will be battery-driven, which will make the power budget tight for all the power-consuming components in a device. This will affect, for instance, CPU processing, memory size/usage, signal processing, and transceiver output/input power.
A central node in a scatternet will in other words be exposed for large demand of power. Central nodes like master/slave switches will rarely be permanently set up, they will be configured dynamically and sometimes we can not set up a better power supply in these devices. Hence, roaming Bluetooth devices will be exposed for a large power consume.

Scatternet Forming and Reforming

Scatternets can also be rearranged to give better overall performance. For instance, if two slave nodes need to communicate, it might be wiser to create a new piconet that solely contains these two nodes. The nodes can still be part of their original piconets if traffic flows to or from them, or if they need to receive control information. Since the frequency-hopping spread-spectrum (FHSS) system makes Bluetooth very robust against interference, new piconets gain substantially more capacity than they lose as a result of increased interference between them.

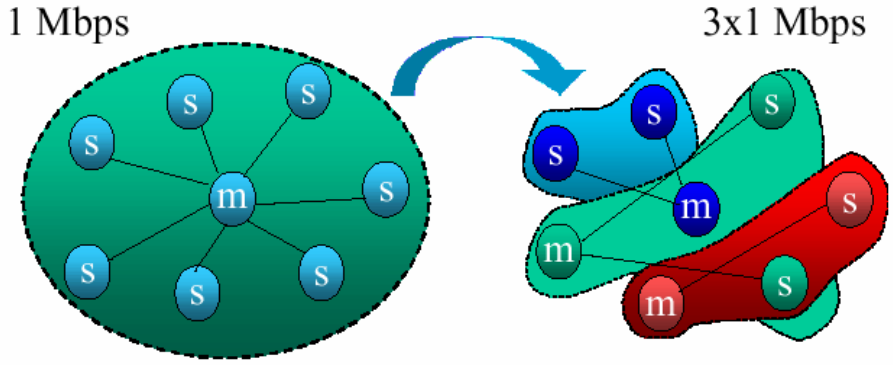


Figure 56: Rearrangement of piconet to improve performance

Another way of describing some of the problems concerning the scatternet issue is the reforming of scatternets. As the figure shows below, two piconets at one certain time could be connected together between two slaves and seconds later between two other nodes. The nodes are still a part of the same scatternet, but the routing tables now have to be quite different.

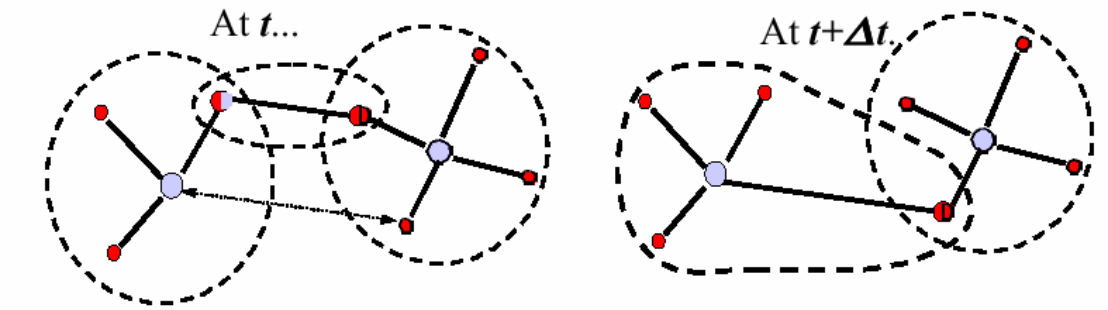


Figure 57: Reforming of scatternets

The Bluetooth devices will at all times try to maintain the “optimal” scatternet configuration. How to configure the connections are proprietary. It will depend on;

- Connectivity
- Traffic distribution
- Mobility and traffic dynamics.(but will a steady state ever be reached?)

Routing in scatternets

The issue of routing packets between any pair of nodes in a PAN becomes a challenging task because the nodes can move randomly within the network. A path that was considered optimal at a given point in time might not work at all a few moments later. Moreover, the stochastic properties of the wireless channels and operating environment add to the uncertainty of path quality.

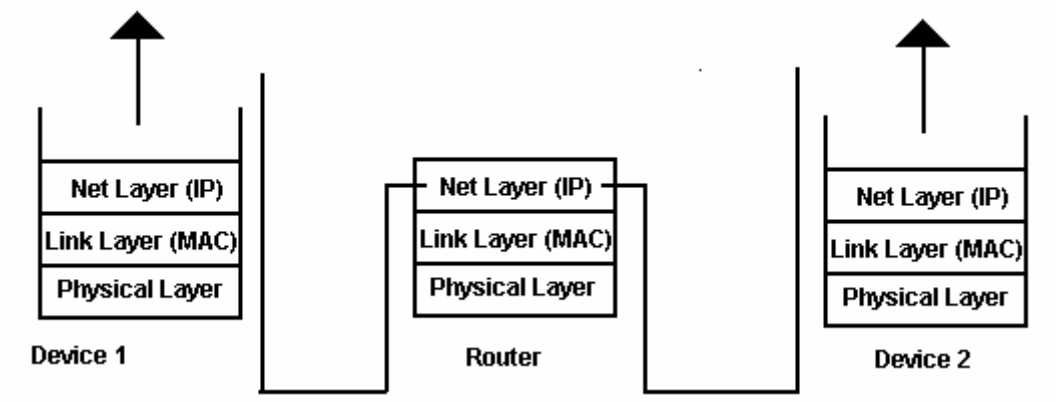


Figure 58: Routers work on the IP layer

Traditional routing protocols are proactive in that they maintain routes to all nodes, including nodes to which no packets are being sent. They react to any change in the topology even if no traffic is affected by the change, and they require periodic control messages to maintain routes to every node in the network. The rate at which these control messages are sent must reflect the dynamics of the network in order to maintain valid routes. Thus, scarce resources such as power and link bandwidth will be used more frequently for control traffic as node mobility increases.
An alternative approach involves establishing reactive routes, which dictates that routes between nodes are determined solely when they are explicitly needed to route packets. This prevents the nodes from updating every possible route in the network, and instead allows them to focus either on routes that are being used, or on routes that are in the process of being set up.
In a simulation study, SwitchLab (Ericsson Research) compared two reactive routing algorithms (ad hoc on-demand distance vector, AODV [8], and dynamic source routing, DSR [9], and one proactive routing algorithm (destination-sequenced distance vector, DSDV [10]). All these protocols are under development within the IETF MANET working group [11]. In every case tested, the reactive algorithms outperformed the proactive algorithm in terms of throughput and delay. Moreover, the reactive protocols behaved similarly in most of the simulated cases. The main conclusion drawn from this study is that a reactive approach might well be necessary in a mobile environment with, such as in a PAN or in interconnected PANs. The proactive approach depletes too many resources updating paths (if the route-update periods are to match the mobility of the nodes). If the update interval is too long, the network will simply contain a large amount of stale routes in the nodes, which results in a significant loss of packets.

The current IP dynamic host configuration protocols (DHCP) and ARP, rely on link layer multicast connectivity. Generally, the protocols will not work beyond an IP router, which means that they will not reach nodes located more than one Bluetooth hop away in an IP-routed scatternet. A scatternet that provides broadcast segment-like

connectivity would enable these protocols to work for Bluetooth-based IP hosts that are separated by multiple hops.
To operate efficiently, the routing function should be joined with the function for forming scatternets. A routing function on the IP layer would thus need to be adapted to, or interact very closely with, the underlying Bluetooth layer, which violates the idea of keeping the IP layer independent of the link layer technology.
Other, non-IP based, applications may use the scatternet functionality provided by the Bluetooth networking layer, i.e. the Bluetooth devices are not forced to host an IP layer to utilize the networking features of the scatternet.

To realize routing in scatternets we have to implement a router in every Bluetooth device, to ensure that every device if we want to ensure that every Bluetooth device can be a master-slave switch. The cost of implement an IP router in every Bluetooth unit can be quite expensive and is a less desirable situation.

The routing is on the other hand a process that requires a lot of processing. In a Bluetooth network were the power and the bandwidth is low, is this also a scenario we will try to avoid.

In summary, the best way of providing networking among the nodes in a Bluetooth scatternet would be to perform the packet forwarding, not routing, on a Bluetooth network layer residing below IP.
As we discussed in the previous chapter BNEP will solve this problem in some ways, but is still not yet solved completely for scatternet. The problem is until now how to solve the address resolution. That has to be taken care of by the Bluetooth layers. In other words we need a scatternet-wide ARP.

Forwarding

This approach has already been discussed in the BNEP evaluation. To brush up, the goal of BNEP was to ensure that we loose the master- slave topology and that every slave will act like peers. In the future BNEP will hopefully offer a broadcast segment infrastructure to the IP layer, similar to Ethernet spanning over an entire scatternet. Today it only works between LAN and piconets.

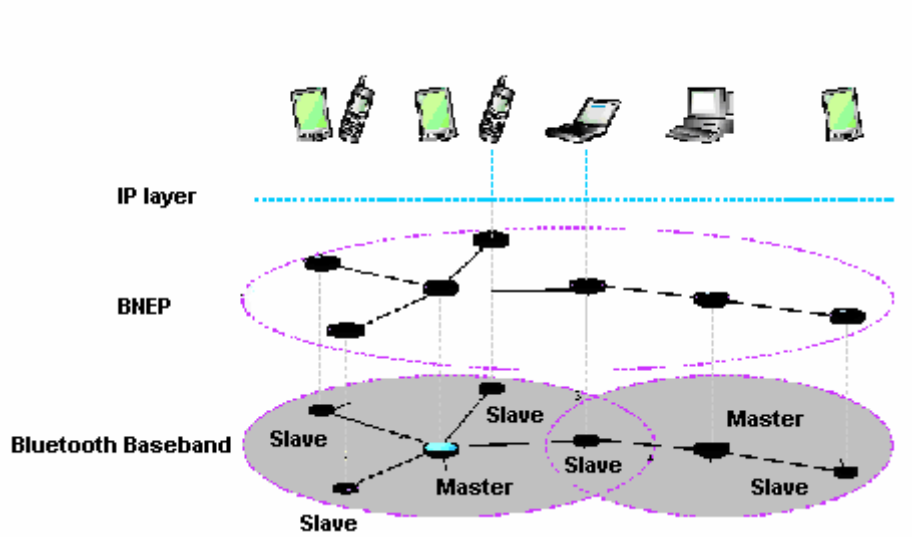


Figure 59: BNEP simulating an Ethernet

Note that IP still has the task of performing routing on the wide area scale and routing can also be performed in network access point. But the BNEP protocol helps us with providing a scalable routing architecture for Bluetooth and Ethernet by forwarding.

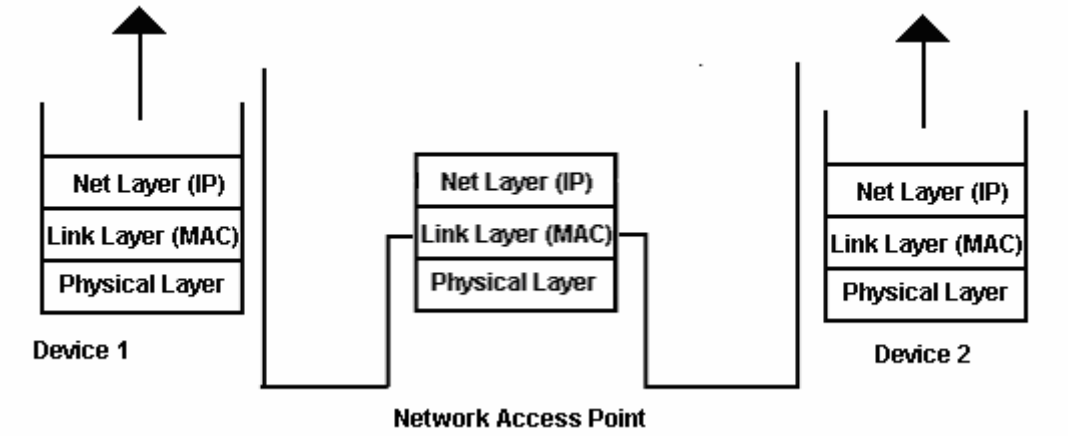


Figure 60: Bridging by means of BNEP on the Link Layer

Summary

As we have pointed out above, there are several issues that indicate that Bluetooth scatternets are making things a lot more complex. We got problems like lower bandwidth, scheduling, high power consumption, forming/reforming and routing problems.
We think that scatternets have too many drawbacks to be a future oriented way to make a larger Bluetooth network. We also think that we must (as long as it is possible) try to avoid routing in Bluetooth devices. These conclusions lead to the BNEP protocol.

The current work in the PAN WG of the Bluetooth SIG focuses on developing IP support based on BNEP, that creates a broadcast segment similar to Ethernet over a whole scatternet. This will enable a straightforward reuse of a number of IP related protocols for configuration and address resolution (such as DHCP, ARP etc.) typically suited for LAN/GPRS access environments, but also for standalone or interconnected PANs.

A larger Bluetooth network should be based on BNEP, but we will prefer a solution without scatternets. The next chapter will suggest our own solution of how we think a larger Bluetooth network implementing IP should look like.

7 The proposal of a large Bluetooth network

Objectives

- How have others solved larger Bluetooth network topologies
- Introduce the BNEP based solution
- Discussion of issues in the solution, how to improve the solution

Contents

Introduction

Our conclusion from the previous chapters (and the thesis) is that BNEP is the best solution for putting IP over Bluetooth. We are now going to suggest how a larger Bluetooth Network can be feasible using BNEP. The possibility exists of implementing a LAN based network of Bluetooth devices, but this still requires an intermediary of a non-Bluetooth link between two LAN Access Points (LAP) devices. Currently there is no profile specified which elaborates how intersecting Piconets can communicate. However this has not prevented research into this area and the follow-on question of roaming. There have been a lot of papers concerning modifications to or analyses of Bluetooth roaming and routing.

- **A Routing Vector Method for Routing in Bluetooth Scatternets** [16]
- **IP Services over Bluetooth: Leading the Way to a New Mobility** [17]
- **Routing Connections In Bluetooth** [18]
- **Transmission of IP Packets over Bluetooth Networks** ***(Note: work in progress)*** [19]

Several of the present solutions are based on routing and different ways of updating the routing tables. Some of these require special made Bluetooth protocols and extra equipment (like servers, nodes etc).This makes Bluetooth become a more expensive solution. One of Bluetooth's main goals was to be cheap. We aim to use the BNEP protocol and make a PAN to PAN solution that is cheap, easy to implement and efficient.

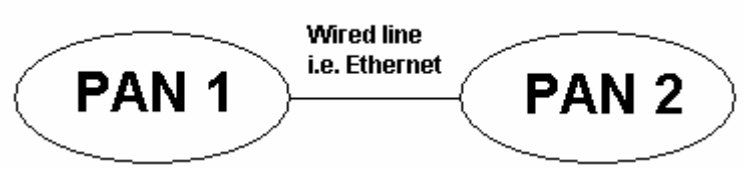


Figure 61: Pan to Pan solution

BNEP will provide a non-router based network, meaning no routing is required between two local PANs within an IP subnet. This agrees with Bluetooth's characterization and main goals of being a cheap and easy solution.
Just as we have discussed in the BNEP chapter, we will try to make a network where we can remove the master slave topology and form a peer network. If we don't use routing we don't have the same requirements for routing update and handover. We will discuss this topic later in this chapter.
BNEP is based on Ethernet infrastructure and within an IP subnet the Bluetooth node will be able to roam randomly without any routing update. In a larger scaled Bluetooth network, we analyse the use of Mobile IP to roam between IP subnets. This points we will also discuss in this chapter.

The proposal of our Bluetooth network.

Below we see the proposal of how we mean a larger Bluetooth network should be realized topologically.

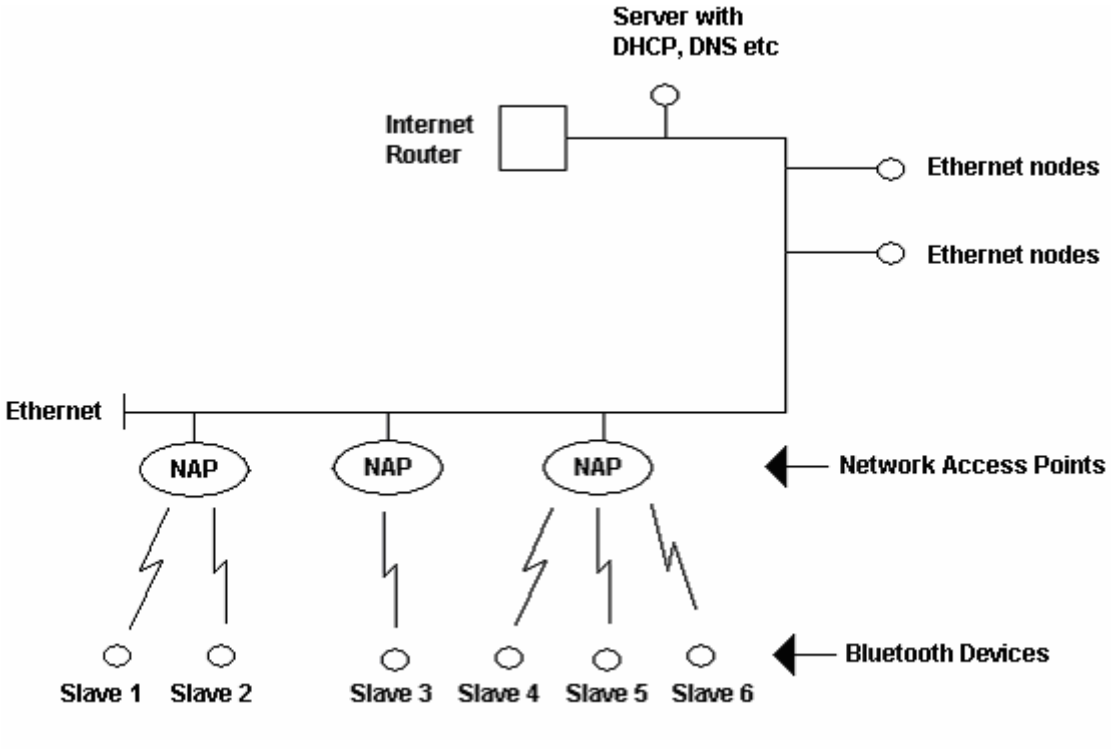


Figure 62: Proposal of how to run a large Bluetooth network with BNEP

The network is scaleable, fairly scheduled and is easily adopted to existing networks like Ethernet. There is nothing particularly new about the network and it is primary based on the PAN profile. That's why this solution may be widely adopted and accepted and this is one of the reasons why BNEP was designed the way it is.
We think that this solution can be used in buildings where they have one or more Network Access point in every room.

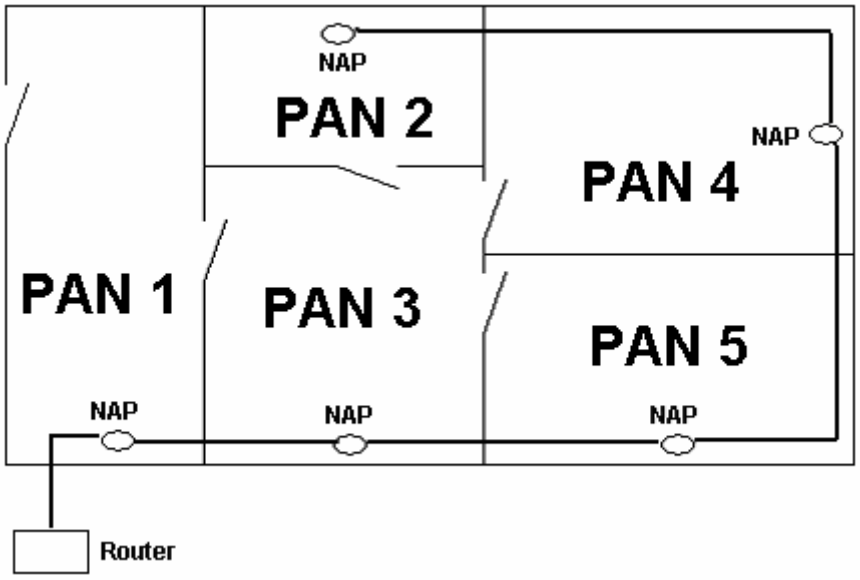


Figure 63: An example of how to build a set of Personal Area Networks

The scenario can be used in several environments with different service requirements. Some scenarios may be in a hospital, personal house, business building etc. The type of service they offer may vary from location based services to pure internet access. The way the network is designed, it has some weaknesses and there is plenty of space for improvements. Following on we will discuss IP issues with this solution.

Issues concerning PAN to PAN with BNEP.

This part of the paper we will discuss some issues concerning Bluetooth and IP. We think the proposed solution is a good solution to run IP over Bluetooth today. Still, the method has some drawbacks and problems. The topics are ARP, DHCP, Mobile IP, scaleability, bandwidth, routing, handoff, roaming, interference etc. We will discuss how it works, if it is good enough and possibly improvements.

ARP – how to send an IP packet

To make it clear how an IP packet is sent between two PANs we decided to make a simple explanation of device A in PAN A sends a packet to device B in PAN B.

- Device A wants to send an IP packet and sends a broadcast ARP request encapsulated in BNEP.
- The NAP receives the BNEP packet containing broadcast ARP, removes the BNEP header, put on an Ethernet header and forwards the ARP to the Ethernet segment as an Ethernet broadcast ARP message.
- The current NAP to Device B receives the ARP broadcast and encapsulates the Ethernet message into a BNEP packet and sends a broadcast ARP in the piconet.
- Device B answers with ARP reply and sends it's hardware Bluetooth address to the device B's hardware Bluetooth address encapsulated in BNEP.
- Device A receives the ARP reply and is ready to send the IP packet on Bluetooth encapsulated in BNEP.
- Device A sends a BNEP packet with device B's BNEP/Bluetooth address as destination.
- NAP A replace the BNEP header with a Ethernet header and sends in on the Ethernet.
- NAP B replace back BNEP header with the Ethernet header and sends the BNEP packet to Device B.

To summarize; the BNEP simulates Ethernet and every NAP will as we have mentioned before, act like a hub and the Bluetooth address will become like MAC addresses in an Ethernet. The solution based on ARP is in some ways great, we then don't need any kind of routing in the access points. But on the other hand we can get a lot of unnecessary broadcast traffic.

BNEP has solved this problem by adding filtering in the access points. This means that the access point don't send every broadcast message and every Ethernet message into the Piconet. So instead of acting like a hub, the network access point more or less acts like a switch or learning bridge. The access point evaluates each packet on link level and decides if it belongs to the Piconet or not.

DHCP

It is desirable for every Bluetooth device that they receive an IP address when they arrive in a Piconet or in other words log on to a network access point. To get a topologically correct IP address, every Bluetooth Device have to get an IP address from the DHCP server on the LAN segment. This is already explained in the BNEP chapter, but we will give a short repetition.
Every time a Bluetooth device connects to a Network Access Point, it will request an IP address from a DHCP server, by sending DHCP-discover and DHCP-request messages to Ethernet through the NAP.

As long as a Bluetooth device is under the same IP subnet, the Bluetooth device should receive the same IP address even if you change the NAP. The DHCP server usually saves your IP address for 12, 24 or 36 hours. This means that the Bluetooth Device should receive the same IP address if it connects to the same subnet within 12 hours. If the Bluetooth Device maintains the same IP address and not receives a new one, we don't need any kind of mobile IP handover and the Ethernet segment will look the same as before.

Roaming

As we mentioned previously roaming within an IP subnet will in most cases be imperceptible to IP (but not TCP) as long as the same IP address is maintained.

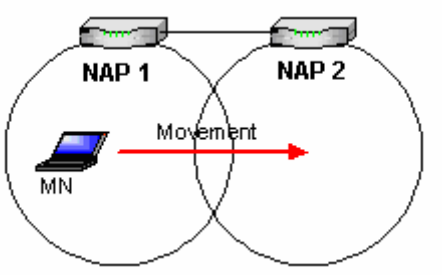


Figure 64: Roaming between to Network Access Points on the same Ethernet segment

The problem arises when we move to another IP subnet or when we are not able to maintain the same IP address within the same IP subnet.

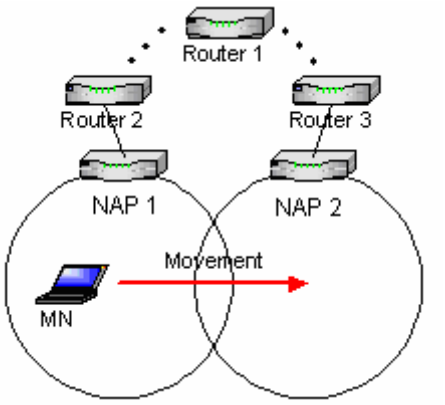


Figure 65: Roaming between to Network Access Points between two Ethernet segments

This problem may be solved by the use of Mobile IP. Mobile IP will be discussed later.

Roaming and it's affect on TCP

We mentioned instead that roaming will have little effect on IP level, both within and between IP subnets. The problem first arises on the TCP layer because of the latency when connecting on the L2CAP level.

When we are roaming in a Bluetooth environment, we have to shut down the Bluetooth connection every time we are leaving an access point and set up a connection to a new access point. This procedure is quite slow and can in worst case exist in 10.24 sec.

Before we can send anything on the BNEP link we have to be connected on L2CAP level. We have described the L2CAP in chapter 2. We need to do Inquiry, paging etc to get connected. The Inquiry procedure is especially slow. This is used to find the identity of the Bluetooth devices in the close range. The discovering unit collects the Bluetooth device addresses and clocks of all units that respond to the inquiry message..

This is not good if you are roaming in a TCP session. If you are loosing to many IP, packets the window size will decrease dramatically. Additionally, if the connection is absent for a longer period of time there is a good possibility to get a TCP timeout and loose the TCP connection.

Due to this, it's very important to get the L2CAP reconnection to go as fast as possible. Unfortunately this is a limitation in Bluetooth and very hard to improve. Still, there are some suggestions like doubling the clock hopping sequence.

We know it is possible for a slave to be connected to different masters at once. If we let the range of each access point overlap each other, it is possible to connect several NAPs from a on a L2CAP level. A Bluetooth device can in other words be able to connect two NAPs at the same time and let some of the connections stay in HOLD mode.

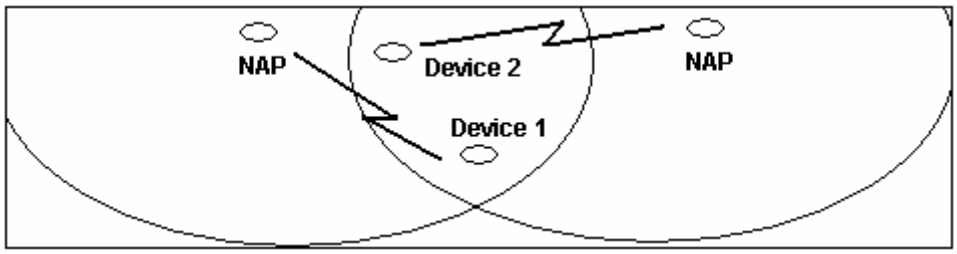


Figure 66: Overlapping Network Access Points, a possibility of less latency

The problem is that we must use a lot of bandwidth to periodically send Inquiry messages to discover new devices and we can't send an inquiry when we are sending data.

Anyhow the BNEP is still not supported in scatternet and situations like this, but is a good proposal of how we can improve the L2CAP connection latency in the future.

Mobile IP

The Internet Protocol is based on the assumption, that each host is assigned a hierarchical IP address: a netID and a hostID. Consequently, IP datagrams to a host are routed to the network given by the netID first, which then takes care of the delivery of the actual host. The netID is the name for a given IP subnet.
The problem arises when a roaming Bluetooth device change to another IP subnet. The packet will then no longer be routed correctly, and the IP address will no longer be valid in the new IPnet.

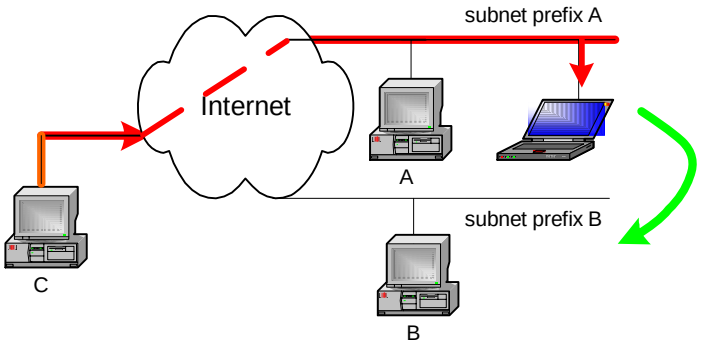


Figure 67: Mobile IP, moving between two IP subnets

Generally Mobile IP has an easy ideology. Every mobile device has a static home IP address that is taken care of a home agent. Whenever a device is coming in to a new router domain it, gets a new local IP address and informs its home agent about the new IP address. IP packets sent to the home agent will then automatically be forwarded to the so called foreign agent and then the mobile device.
IPv6 and IPv4 have two different ways to run Mobile IP, but their differences have little influence of our description of Mobile IP in general. Never the less Mobile IPv6 has a more efficient way to run Mobile IP it is faster and better than MIPv4.

Handoff/handover

In certain scenarios when a Bluetooth is in motion the Bluetooth device must support handoff or handover to maintain existing connection while Bluetooth devices move from one IP subnet to another.
Bluetooth is a technology with minor condition of bandwidth and realtime support and the need for a seamless handover desirable but not absolutely necessary.
We can do handover on different layers; Bluetooth lower layer, BNEP handover and Mobile IP handover.

Bluetooth handover/handoff

Bluetooth (as it currently stands) does not support handover/handoff between Bluetooth piconets. If a device is in danger of loosing its link to one master, no provision is made to transfer it to another master.
However, although it is not defined in the spec, we personally think it is possible but it will be vendor-dependent which might have problems in interoperability.
Handover/Handoff might be useful with LAN Access Points. In this case roaming, hand-off might be useful for having a seamless connection.

The main problem is in fact the lack of Received Strength Signal Indicator (RSSI). Normally most wireless system has this kind of variable to decide when to do a handover or handoff. In Bluetooth we don't have this kind of method to measure the strength of the radio signal, but there have been suggestion using piggyback RSSI or Bit Error Rate (BER) indication.
Piggyback RSSI is RSSI measured by the access point, and the access point decides when to do handover. BER indication is a method for measuring when the error level of transmissions is so high that we perceive that it is time to do handover.

The reason we prefer to do the handover on this layer is shorter latency. If we can make 2 Bluetooth devices exchange information on a Bluetooth level we might lose the Inquiry and Paging procedures. Until this point there is no method to solve this between 2 access points, but we can catch a glimpse of a solution with BNEP control messages of lower Bluetooth information.

BNEP handover/handoff control messages
Currently BNEP does not support any kind of special handoff/handover mechanisms, but we think BNEP in the future could have some kind of handoff/handover functionality.
The BNEP protocol has possibilities to be further extended. One of the extensions may be handoff/handover control messages between Network Access points and some Mobile IP handover triggers to higher layers like IP.

Mobile IP handover
Handover in Mobile IP will happen every time a mobile node roams between two IP subnets.
This service has a long latency and will cause QoS reduction.

The Mobile IP handover consists of three steps:

- Detection of a new network
- Get a new local IP address (so called CoA)
- Location update to the Home Agent

Usually the Bluetooth node will notice it has arrived into a new network, when it receives agent advertisements from the local router. These messages are sent periodically and will help the node to be aware of if it has lost contact with the net, found a new net or not connected.
When the node is aware of coming into a new network it will send a location update (binding update).
In our network scenario, we will not need the same agent advertisement. In despite of still having agent advertisement, we may use lower Bluetooth link information to be able to do more efficient handovers.
As we already have pointed out before, when Bluetooth devices connect to a new access point it will automatically ask for a new IP address and do the location update without use of agent advertisements.
Whether to use Mobile IPv4 or Mobile IPv6 is regardless from our point of view. MIPv6 will make a faster handover, but they both have the same main functionality.

Routing

The solution will not need any special new solution with the involved routers. As long as we are using mobile IP the routing tables will not need to be changed at all. In other words we no longer need any servers or registers except from the mobile IP support.

Scaleability

The Internet Protocol is the key tool used today to build scalable, heterogeneous internetworks. One way to think of IP is that it runs on all the nodes in a collection of networks and defines the infrastructure that allows these nodes and networks to function as a single logical internetwork. Data from applications and higher-level protocols are encapsulated in IP datagram packets in order to traverse the networks. Ethernet and BNEP accomplish this ideology. An obvious difference is the bandwidth between Ethernet and Bluetooth BNEP, but the Bluetooth devices will be fair scheduled on the same level as Ethernet nodes.
The problem is that in a piconet it can't exist more than 7 slaves. A Piconet can in other words only have 7 devices. If there are any further devices trying to connect, they will just be rejected. But do we really need more than 7 device support within a NAP?
We think that normally there will quite seldom be more than 7 devices sending at the same time to one access point in one room. In some scenarios it could be a problem, and then we suggest that it should be inserted another access point in the same area.

Bandwidth

Ethernet is based on multiple access and has a lot of broadcast messages. BNEP makes Bluetooth devices become peers in an Ethernet segment and therefore the Bluetooth devices originally will receive all Ethernet broadcast messages. Bluetooth has originally a low bandwidth and is not suited for this kind of communication and it is desirable to spend as less unnecessary bandwidth as possible.
Bluetooth is not a technology with high bandwidth, but we want to get the bandwidth performance as high as possible. To spare the Bluetooth network for this, the network access point may implement filter functionality. It will filter only the Ethernet packets that are needed within the Bluetooth net.

Summary

We think that at the moment this is a good and feasible proposal to make a large Bluetooth network. The most important things are that it is cheap and easily deployed. Bluetooth has some roaming latency problems, that is hard to overcome. There are solutions that solve these problems in Bluetooth, but we think they are far too complicated, expensive and have no future on the marked.
The most important thing about this solution is to design the network properly. If you want to roam between several PAN in a building, you have to design the network in a way that you overcome routing problems. If interconnected PAN exists on the same Ethernet segment, we don't need to do any handover or binding update. The Bluetooth device will mostly keep the same IP address and will still be a part of the same multiple access Ethernet segment.
Our main problem is the roaming when you are in a TCP session. The Bluetooth device spends to long time to reconnect on L2CAP level and the chance of getting a TCP time out is possible. We think this is something we need to accept. Bluetooth is not designed to do a large TCP session which requires a lot of bandwidth and Bluetooth is designed to be cheap and easy to use. If you are roaming and lose your TCP connection once in a while, we think this would be acceptable. In our proposal Bluetooth is still easy and cheap. In a scenario where you don't require high bandwidth, this should be a very good solution.

8 Thesis Conclusion

The thesis has focused on several aspects of running IP over Bluetooth. Initially our main goal was to find the best way to run IP over Bluetooth and it has been our main goal during the project.

Bluetooth is a technology that has an easy and wide area of employment and it has different requirements for different environments. The “old” way to run IP over Bluetooth is still a solution in some scenarios like dialup connections etc and LAN access profile, but we have pointed out some difficulties. Some of the problems when we are running IP over RFCOMM are; PPP and ad-hoc network support, the need of a PPP server and a lot of overhead, latency in connecting and a need of routers in every access point.

We have also made a short evaluation of another method with IP over L2CAP. This method is more efficient but it still requires routing in every access point and we would still have the master-slave topology which we are trying to avoid.

The solution to most of the problem above is solved by the PAN profile and the BNEP protocol. In BNEP the access points don’t need to act like routers, instead an access point will simulate an Ethernet hub. The BNEP protocol will also remove the master slave topology and every node will look like peers. The protocol will support broadcast and every common related internet protocols like ARP, DHCP and DNS. However, the BNEP protocol is not yet totally specified. Some of the problems we discussed were handover/handoff handling and scatternets issues, because up to now BNEP only supports Piconets.

Due to this Scatternet problem, we discussed some of the issues concerning Scatternets in the preceding two chapters and how IP over Scatternets will make the network more complex. Some of the problems are lower bandwidth, possible routing problems, reforming problems etc. Hence we concluded that a large Bluetooth network should be based only on Piconets using BNEP. The current solution is exhibited in the end. The solution follows the ideology of Bluetooth that it be a cheap and easily deployed technology. The most important thing is to design the network properly. To overcome roaming problems the PANs you may roam in have to exist in the IP subnet. As long as we are roaming within the IP subnet, we don’t need to update routing tables or to do location update. Procedures like Mobile IP location update are resource expensive and slow and it is a goal to decrease the number of location updates as low as possible. But roaming between IP subnets may be solved by means of advanced mobile IP.

The main problem is the reconnection on the L2CAP level. The latency is too long and we can get problems with TCP timeout. In spite of this we believe that this could be a suitable solution for low cost Bluetooth devices. Normally, we believe people will not roam very often in a TCP session and we think most people will be prepared to wait some seconds extra due to the switching access point, if the service is low cost.

In the future we think a lot of work will be focused around the BNEP protocol. At the moment work is going on to realize Satternet support for BNEP and IP and should

appear in the next draft of BNEP. We can also imagine some handover capabilities supported by BNEP. It should be possible to implement BNEP and Ethernet control messages to make a controlled handover between two access points.

9 References

[1] Bluetooth Special Interest Group (SIG), http://www.Bluetooth.org

[2] Bluetooth Connect Without Cables 1.1 Second edition, Jennifer Bray and Charles F Sturman. ISBN 0-13-066106-6

[3] Bluetooth Special Interest Group, “Bluetooth Personal Area Networking Profiles”, Specification of the Bluetooth System, Version 0.95, June 26, 2001

[4] Bluetooth Special Interest Group, “Bluetooth Core”, Specification of the Bluetooth System, Version 1.1, February 22, 2001

[5] Internet Engineering Task Force, “The Point to Point Protocol”, RFC1661, July 1994

[6] Internet Engineering Task Force, “PPP in HDLC-like Framing”, RFC1662, July 1994

[7] Bluetooth Special Interest Group, “Bluetooth Network Encapsulation Protocol (BNEP) Specification”, Specification BNEP Protocol, Version 0.95a, June 12, 2001

[8] Charles E. Perkins, "Ad Hoc On Demand Distance Vector (AODV) Routing." Internet draft, draft-ietf-manet-aodv-02.txt, November 1998.

[9] Josh Broch, David B. Johnsson, David A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad hoc networks." Internet Draft, draft-ietf-manet-dsr-00.txt, March 1998.

[10] Charles E. Perkins and Pravin Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers." In Proceedings of the SIGCOM '94 Conference on Communications Architecture, protocols and Applications, pages 234-244, August 1994. A revised version of the paper is available from http://www.cs.umd.edu/projects/mcml/papers/Sigcomm94.ps. (1998-11-29)

[11] Mobile Ad hoc Networks (MANET). URL: http://www.ietf.org/html.charters/manet-charter.html. (2000-05-28).

[12] http://www.iana.org/assignments/ethernet-numbers

[13] “The Ethernet – A Local Area Network”, Version 1.0 Digital Equipment Corporation, Intel Corporation, Xerox Corporation. September 1980.

[14] Internet Engineering Task Force, “Dynamic Host Configuration Protocol”, RFC2131, March 1997

[15] Internet Engineering Task Force, “An Ethernet Address Resolution Protocol”, RFC826, November 1982.

[16] P. Bhagwat, A. Segall, A Routing Vector Method for Routing in Bluetooth Scatternets, 1999
http://www.palowireless.com/infotooth/documents/RVM_for_Routing_in_Bluetooth_Scatternets.zip

[17] M. Albrecht, M. Frank, P. Martini, A. Wenzel, M. Schetelig, A. Vilavaara, IP Services over Bluetooth: Leading the Way to a New Mobility, 1999
http://www.palowireless.com/infotooth/documents/lcn99-bt1.zip

[18] Mc Daid C., Routing Connections in Bluetooth, 2000-04-25
http://www.palowireless.com/infotooth/documents/Routing_Connections_in_Bluetooth.zip

[19] Atwal K. & Akers R. , Transmission of IP Packets over Bluetooth Networks (work in progress), 2001-05-27 http://www.globecom.net/ietf/draft/draft-akers-atwal-btooth-01.html

Acronyms and abbreviations

The list of abbreviations is necessary for understanding IP over Bluetooth.

Abbreviation or Acronym	**Meaning (Reference page)**
ACL	Asynchronious Connection Less(21)
AODV	Ad-hoc on demand distance vector (87)
ARP	Address Resolution Protocol (72)
BB	Baseband (20)
BNEP	Bluetooth Network Encapsulation Protocol (68)
CID	Connection Identifier (34)
CL	Connectionless
CoA	Co-located care of address (99)
CoD	Class of Device
CTS	Clear to send (37)
DAC	Device Access Code (25)
DLCI	Data Link Connection Identifier (37)
DSDV	Destination sequenced distance vector(87)
DSR	Dynamic Source Routing (87)
DT	Data Terminal (45)
FA	Foreign Agent (99)
FH-CDMA	Frequency Hopping Code Division Multiple Access (22)
FHS	
FHSS	Frequency Hopping Spread Spectrum (86)
GFSK	Gaussion Frequensy Shift Keying (22)
GIAC	General Inquiry Access Code (24,26)
GN	General Ad-hoc Network (65)
HA	Home Agent (99)
HCI	Host Controller Interface(31)
IP	Internet Protocol
IPCP	IP control protocol (52)
L2CAP	Link Layer Control and adaption layer Protocol (34)
LAN	Local Area Network (2,71)
LAP	LAN access point (45)
LC	Link Controller (20,23,28)
LC	Logical Channel(22)
LM	Link Manager (16,20,28)
LM	Link Manager(29)

LMP	Link Manager Protocol (28)
LMP	Link Manager Protocol (29)
M2M	Machine to Machine (8)
ME	Management Entity (65)
MTU	Maximum Transmission Unit
MTU	Maximum Transmission Unit
NAP	Network Access Point (49,67)
NCP	Network control Protocol (52)
OSI	Open Systems Interconnect (model)
PAN	Personal Area Network (67)
PANU	Personal Ad-hoc Network User (66,67,73)
PDU	Protocol Data Unit (38)
PPP	Point to Point Protocol (48)
PSM	Protocol Service Protocol (62)
QoS	Quality of Service (28,45)
RFCOMM	36
RTS	Ready to Send (37)
SAR	Segmentation and Reassambly (35)
SCO	Synchronious Connection Oriented(21)
SDP	Service Discover Protocol(38)
SIG	Bluetooth Spesial Interest Group (14)
TDD	Time Division Duplex(22)
UUID	Service Class Identifier (39)

Table 13: Abbreviation or Acronym

List of Tables

List of Figures