PROJECT REPORT

ON

“BLUETOOTH BASED SENSOR NETWORK”

UNDER THE

GUIDANCE OF

Dr. ANAND KUMAR SOURABH

CSE (IOT) ENGINEERING DEPARTMENT

A.C. PATIL COLLEGE OF ENGINEERING

2022-23

SIGNATURE:

DATE:

REMARK:

PROJECT REPORT

ON

"BLUETOOTH BASED SENSOR NETWORK"

Submitted to:

Dr. ANAND KUMAR SOURABH

Project By:

SR NO	Name	PRN NO	Roll number	Class
1.	PRIYUSH KHOBRAGADE	211112018	52	T. E.
2.	AYUSHI MASKE	201111002	15	T. E.
3.	KAUSTUBH DHAKATE	201111004	07	T. E.
4.	AKSHAY LOHAR	211112032	54	T. E.

BONAFIDE CERTIFICATE

Certified that this report **"BLUETOOTH BASED SONSOR NETWORK"** is the bonafide work of "**PRIYUSH KHOBRAGADE, AKSHAY LOHAR, AYUSHI MASKE And KAUSTUBH DHAKATE"** who carried out the project work under my supervision.

SIGNATURE

HEAD OF THE DEPARTMENT

Professor,
Dept. of Computer Science And Engineering,
AC Patil Engineering College,
New -Mumbai , Kharghar 410210

SIGNATURE

SUPERVISOR

Assistant Professor,
Dept. of Computer Science and Engineering (IOT) ,
New Mumbai , Kharghar 410210

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We affirm that the title “BLUETOOTH BASED SONSOR NETWORK”. Begin submitted in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering is the original work carried out by me. It has not forwarded the part of any other thesis submitted for award of any degree or diploma, either in this or any other University.

Signature of the candidate: -

PRIYUSH KHOBRAGADE	52
AKSHAY LOHAR	54
KAUSTUBH	07
AYUSHI MASKE	15

I certify that the declaration
made above by the candidate is true.

Signature of the Guide,
(Dr. Anand Kumar Sourabh)
CSE(IOT), Assistant Professor**,**
Dept. of Computer Science and Engineering,
AC PATIL ENGINEERING COLLAGE
KHARGHAR – 410210.

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to their guidance and constant supervision as well as for providing necessary information regarding the project x also for their support in completing the project. I would like to express my gratitude towards my parents x my team members for their kindco-operation and encouragement which help me in completion of this project.

I would like to express my special gratitude and thanks to my faculties for giving me such attention and time. My thanks and appreciations also go to my colleague in developing the project and peoplewho have willingly helped me out with their abilities.

TABLE OF CONTENT

BLUETOOTH BASED SMART SENSOR NETWORKS

Abstract:

Wireless sensor networks are networks of small computers, fitted with sensors, microprocessors and wireless interfaces. This technology has achieved a lot of attention lately. For these networks are suggested the wide range of modern and fascinating applications, from personal health care to environmental monitoring and military applications. Different wireless technologies, such as simple RF, Bluetooth, UWB or infrared, may be used to communicate between sensors. The core concepts, features and problems of Bluetooth-based wireless sensor networks are outlined in this, as well as the execution of a simple Bluetooth-based sensor network. The presentation of main problems experienced during the implementation and applied solutions are also done. This is specified on the smart sensor networks using the Bluetooth topology. How smart sensor networks are used and can be implemented using the Bluetooth technology. How they are used as the purpose of communication in industrial field, how they are built, their working and concept is reviewed. Architecture, network, applications and working has been reviewed basically for the communication and research purpose.

Wireless sensor networks are networks of small computers, fitted with sensors, microprocessors, and wireless interfaces. This technology has achieved a lot of attention lately. These networks are suggested for the wide range of modern and fascinating applications, from personal health care to environmental monitoring and military applications. Different wireless technologies, such as simple RF, Bluetooth, UWB or infrared, may be used to communicate between sensors. The core concepts, features, and problems of Bluetooth-based wireless sensor networks are outlined in this, as well as the execution of a simple Bluetooth-based sensor network. The presentation of the main problems experienced during the implementation and applied solutions is also done. This is specified on the smart sensor networks using the Bluetooth topology

Keywords: Wireless sensor networks, Media access control, Wide area network

CHAPTER 1

INTRODUCTION

Wireless sensor networks are networks of small devices having sensors, microprocessors and wireless communication interfaces. This technology has become famous lately. For the purpose of communication in industrial field, WSN technology is widely used. Sensors are used for the communication in industries. In this process, the signals are sent through the wires from each field devices and are monitored on central control room. With the beginning of wiring concept, the field device is used for minimizing the wiring cost. Wireless technology is introduced to eliminate the wires as they are costly, bulky and can be easily damaged.

Ericsson Mobile communications started research to explore the usefulness of a consuming low power, low cost, low ratio interface, and to find a process to remove wires between the devices in the year 1994. Then the Bluetooth technology was invented by an electrical engineer Dr. Jaap Haartsen and named the technology Bluetooth to the honor of the 10^{th} century king Harald "Blue tooth" of Denmark. The aim of Bluetooth is harmony and unification [6]. It also enables the different devices to communicate through wireless connectivity. Bluetooth uses frequency hopping spread spectrum technique and works in the illicit ISM band at 2.4 GHz frequency. A distinctive quality Bluetooth device holds a range of about 10 meters and can be extended to 100 meters. The total bandwidth of 1 Mb / sec is supported by communication channel. A topmost data transfer rate of 721 KBPS maximum of three channels is supported through a single channel.

- Bluetooth is wireless high speed data transfer technology overa short range (10 − 100 meters).
- Bluetooth Wireless Technology (BWT) was developed in 1994at Ericsson in Sweden.
- Purpose − Originally it was build to eliminate the need forcable connections between PDAs and notebook PCs. Later the goals were to enable different devices through a commonly accepted standard for wireless connectivity.
- Ericsson on advent of BWT conceptualized a Radio Technologythrough a wireless personal area network (WPAN). Group called Bluetooth Special Interest Group (SIG) wasformed in 1998 to develop the standard of IEEE 802.1 . This specification standardized the Bluetooth technology world wide.

1.1 BLUETOOTH SPECIFICATIONS :-

1. Developed by: Jaap Haartsen and Sven Mattisson in Sweden
2. Standard: IEEE 802.15
3. ISM Band Frequency: 2.4 GHz
4. Range: 10 − 100 meters
5. Channel Bandwidth: 1 Mbps
6. Maximum Asymmetric Data Transfer Rate: 721 Kbps

Why the name BLUETOOTH ?

The name was adopted as a tribute to the tenth−century Viking king Harald Blåtand who peacefully united Denmark and Norway. Harald liked to eat blueberries, whichgave his teeth the coloration that lead to the nickname "Bluetooth."

TYPE'S OF BWT:

Depending on the *power consumption* and *range* of the device,there are 3 Bluetooth Classes as:

- **Class 1**: Max Power − 100mW ; Range − 100 m
- **Class 2**: Max Power − 2.5mW ; Range − 10 m
- **Class 3**: Max Power − 1mW; Range − 1 m

1.2 BLUETOOTH OPERATIONS :

- BWT− enabled devices operate in 2.4GHz ISM Band (Industrial,Science, Medical band).
- It uses 79 1−MHz frequencies in the ISM Band.
- Technique used − *frequency hopping,* to minimize interference from other networks that also use ISM Band.

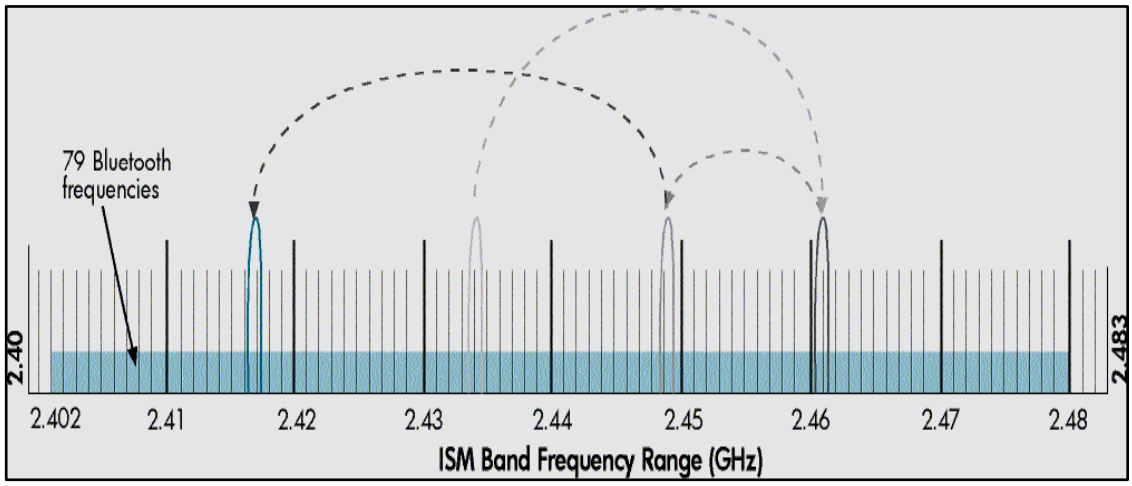


Figure No. 1 BWT−enabled devices hop between frequencies up to 1600 times per second

CHAPTER 2
BLUETOOTH TOPOLOGY

Depending on the type of connections established between Bluetooth devices, 2 main topologies are as:

1. PICONET TOPOLOGY, and
2. SCATTERNET TOPOLOGY

2.1 A Piconet:

A *piconet* consists of upto 8 BWT−enabled devices.

When *piconet* is established, one device sets up *frequency− hopping pattern* and other devices synchronize their signals tothe same pattern.

Primary Devices: Those devices which sets the frequency− hopping pattern.

Secondary Devices: Those devices which get synchronized. Each *piconet* has a different frequency−hopping pattern. In Bluetooth, each *piconet* has 1 Master for establishment of *piconet,* and up to 7 Slave devices. Master's Bluetooth address is used for defining frequency−hopping sequence. Slave devices use master clock to synchronize their clocks soas to hop simultaneously. For establishing *piconet*, other Bluetooth devices in range are discovered by an inquiry procedure.

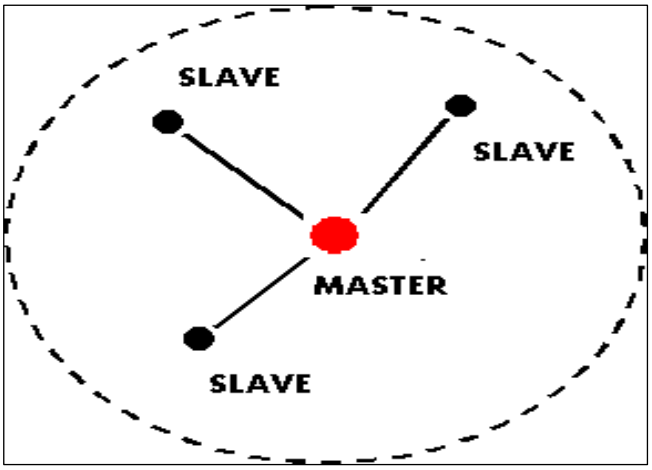


FIGURE 2.1

It is an improvised network used to link wireless devices using Bluetooth technology. It is group of up to 8 devices that shares similar frequencies. It uses the concept of master and slave. Each piconet has one master and rest of the device acts as slave. Usually, the device that starts off the piconet behaves as master. To establish a piconet, first the device searches for the other Bluetooth devices in the range. When the two devices have same frequency then required information is exchanged and to establish the connection paging procedure can be used. For more than 7 devices need to exchange information, there are two possibilities:

The initial one is by putting one or more devices into park state. The modes used in Bluetooth are sniff, hold and park which are used for low power consumptions. When a device changes to the park mode then it disconnects from Piconet, but schedule adjustment will be maintained. The master of piconet continuously transmits signals to invite the slaves to retain the piconet. If there are less than 7 slave devices in the piconet then only the slave will rejoin. If not then one of the active slave devices will be park by the master. Due to these actions, there will be delay and it can be undesirable for some applications like procedure control applications that need instant reply from the command center.

2.2 BTNODES

The BTnodes were developed by ETH Zurich in the con- text of the Smart-Its project . They are based on the Atmel ATmega128L microcontroller - an 8 bit microcon- troller (MCU) clocked at 7.4 MHz, with 4 KiB[1] on chip memory and an external memory chip of up to 64 KiB. The MCU has digital and analog I/O ports that can be used to connect external sensor devices through Molex plugs on the edge of the board. The nodes are equipped with a Bluetooth module (Ericsson ROK 101 007) together with an onboard antenna. Two UARTs connect the MCU with the embedded Bluetooth chip and one of the Molex plugs. Four leds can be used for debugging purposes. The board also contains a voltage regulator: the BTnode can be plugged to power supplies ranging from 3.3 V to 12 V.

MCU	Atmel ATmega128L at 7.372 MHz
Memory	Built in: 128 KiB Flash, 4 KiB SRAM 4 KiB EEPROM External: 64 KiB RAM
I/O	8 Channel 10-bit A/D-converter 2 programmable serial UARTS
Embedded Radio	Ericsson ROK 101 007
External Radio	BTTester (Ericsson ROK 101 007)

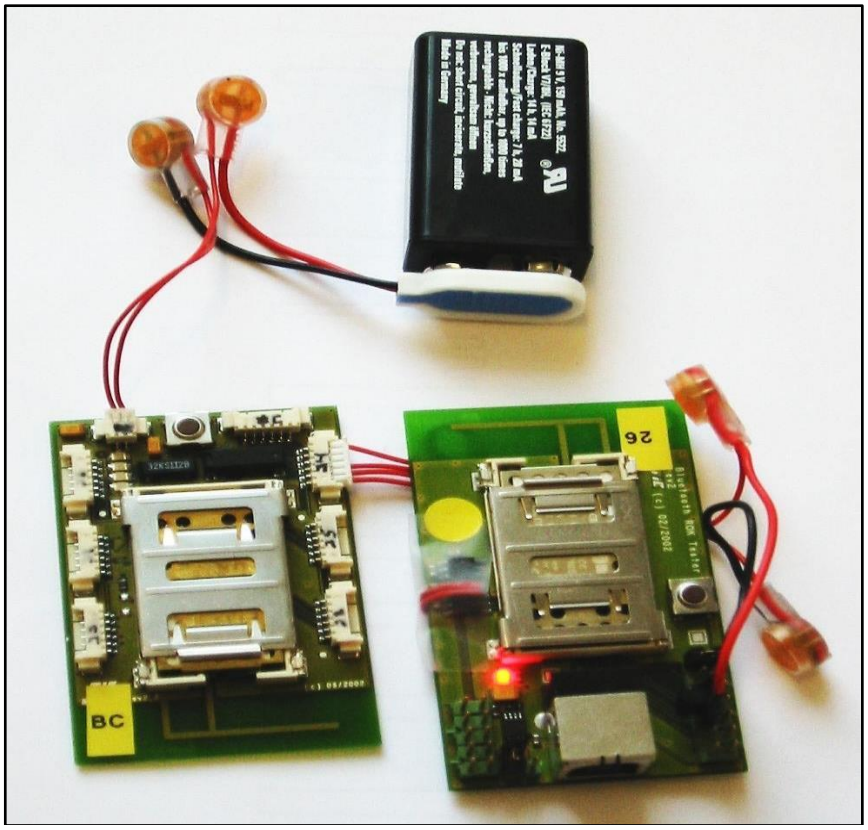


Figure 2.2 : BTnodes characteristics

As explained in the Introduction, we used dual-radio nodes for our experiments in order to assemble a multihop network.[1]EIC standard 60027-2 defines KiB as 1024 bytes.

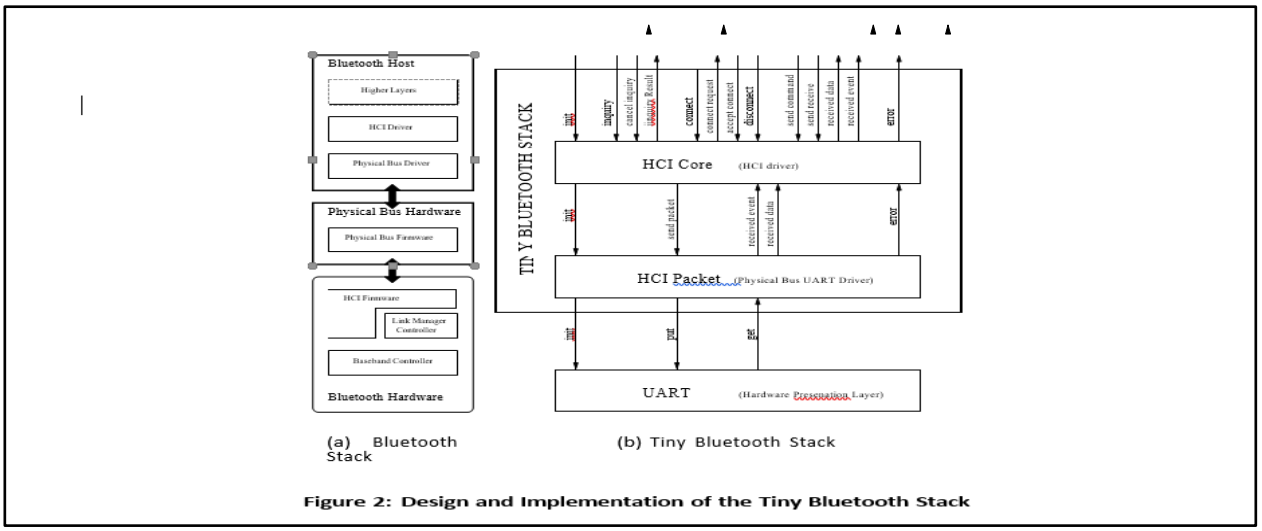


Figure 2: Design and Implementation of the Tiny Bluetooth Stack

FIGURE 2.3

We thus connected an extra Bluetooth module to the BT-node (via the Molex plug connected to an UART). For this purpose, we used the BT tester (also from ETH Zurich),a serial dongle based on the ROK 101 007 Ericsson chip (in- clouding an onboard antenna and a monitoring led).

Figure 1 summarizes the characteristics of the BTnodes. Further documentation can be obtained on the BT node project page.

The Bluetooth specification defines two software layers abstracting the higher layers from the hardware character.

2.3 Physical Bus Driver:

This layer abstracts the characteristics of the physical bus; on the BTnodes the UART connecting the MCU and the Bluetooth radio.

HCI Driver:

This layer maps the interface of the underlying HCI layer (the upper layer embedded on the Bluetooth radio) into the programming model used for implementing the higher layers.

2.4 TINY BLUETOOTH STACK

In this section, we report on the design and implementation of our Tiny Bluetooth stack. We compare it to the ETH Bluetooth stack for the BTnodes and to existing communication stacks developed for TinyOS.

2.6 Design

Our goal with the Tiny Bluetooth stack was to make it possible for TinyOS programs[2] to access a Bluetooth radio.[2]We could reuse most existing components of TinyOS except of course those abstracting hardware elements differentiating the BTnodes from the Mica and Rene motes, e.g., the u arts, the lads, and the clock.

We are not interested in implementing a full-fledged Blue- tooth stack: while the three lower timing sensitive layers are embedded within the radio modules and thus a given, the higher computation intensive layers (l2cap, sdp, profiles) are essentially dealing with the connection of heterogeneous devices (e.g., a mobile phone and a headset) without providing solutions to problems such as cross layer optimizations or multihop routing, which characterize the sensor network regime. The ability to connect to heterogenous devices is not of paramount interest in sensor networks, which are composed of sensor nodes tailored to operate together as one unit. We thus decided not to implement those three higher layers. Rather, we focus the Tiny Bluetooth stack on the interface to the Bluetooth hardware and its mapping to TinyOS. Network assembly and multihop routing are implemented on top of the Tiny Bluetooth stack.

The Bluetooth specification distinguishes the *Bluetooth host*, on which the three upper layers of the protocol are implemented from the *Bluetooth hardware* on which the three lower layers are implemented. The host and the hardware are connected via a physical bus (e.g., USB, UART, RS232). Figure 2(a) illustrates the connection of the Bluetooth host and the Bluetooth hardware.

Our design is based on these hardware abstraction layers. The Tiny Bluetooth stack is composed of two components: *HCI Packet* that corresponds to the physical bus driver and *HCI Core* that corresponds to the HCI driver. The interfaces of both components are described in Figure 2(b) (we omit the events associated to each TinyOS command according to the asynchronous programming model).

TinyOS applications access the HCI Core component to communicate via Bluetooth. The HCI Core component sup- ports the main HCI commands related to (i) the initialization of the Bluetooth radio, (ii) the inquiries used for device discovery, (iii) the connection establishment, (iv) the trans- mission of data and (v) the reporting of errors.

It relies on the services of the HCI Packet layer that maps send commands and receive events onto the UART component detail below the implementation of these components.

Piconet Topology (Modes of Bluetooth Communication) When more than 7 devices need to communicate, then one ormore devices are put in park state.

- Bluetooth Low Power Modes are: *SNIFF, HOLD* and *PARK.*
- Park Mode: A devices disassociates from *piconet* when in parkmode.
- The master consistently sends warnings to invite a slave torejoin the *piconet.*
- The slaves can rejoin only if there a less than 7 slaves.

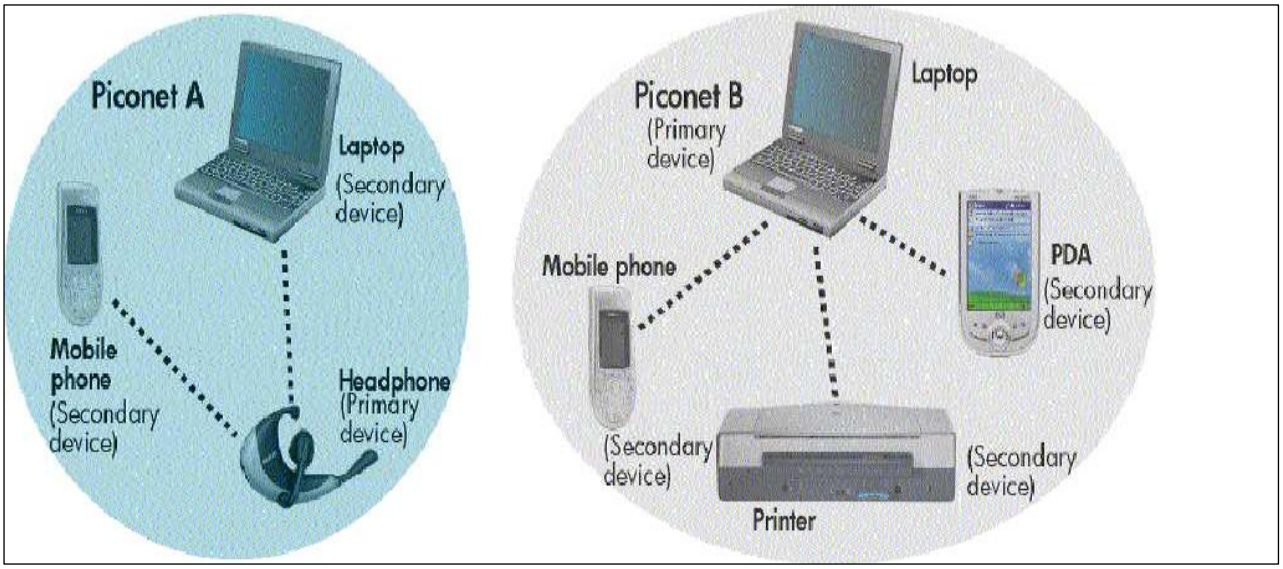


FIGURE 2.6

2.6 Scatternet :-

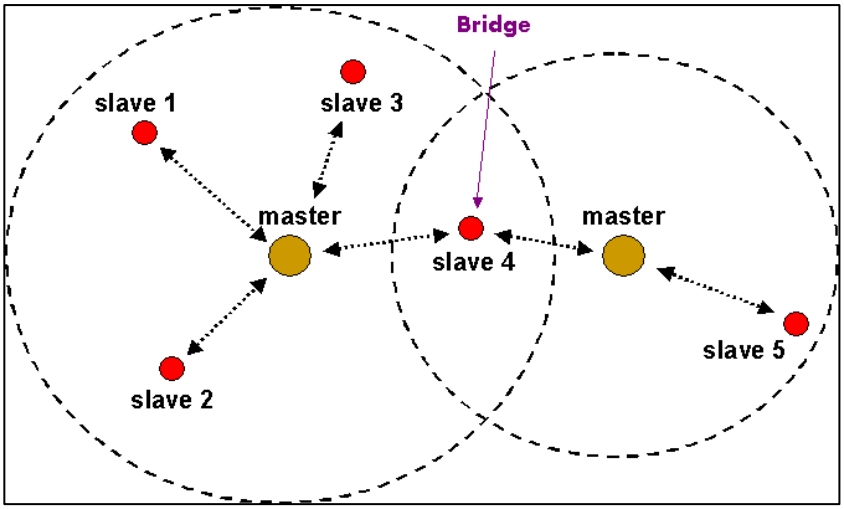


FIGURE 2.6

It is composed of interconnected piconets that maintain communication between more than 8 devices. Scatternets are formed when a device of one piconet that can be master or slave choose to act as a slave in second another piconet. Scatternets provide higher throughput. Also, in different piconets multiple-hop routing between devices is possible. That means at one time only single component can interface in one piconet so they hop from one to another relying upon the channel limit.

Scatternet consists of several *piconets* connected by devices participating in multiple *piconet.* Here, devices can be slaves in all *piconets* or master in one *piconet* and slave in other *piconets.* There is a 'BRIDGE' connecting 2 *piconets* which is also a slavein individual *piconets.*

Advantages of *Scatternet*:

- Higher throughput
- Multi-hope connection between in different piconets.

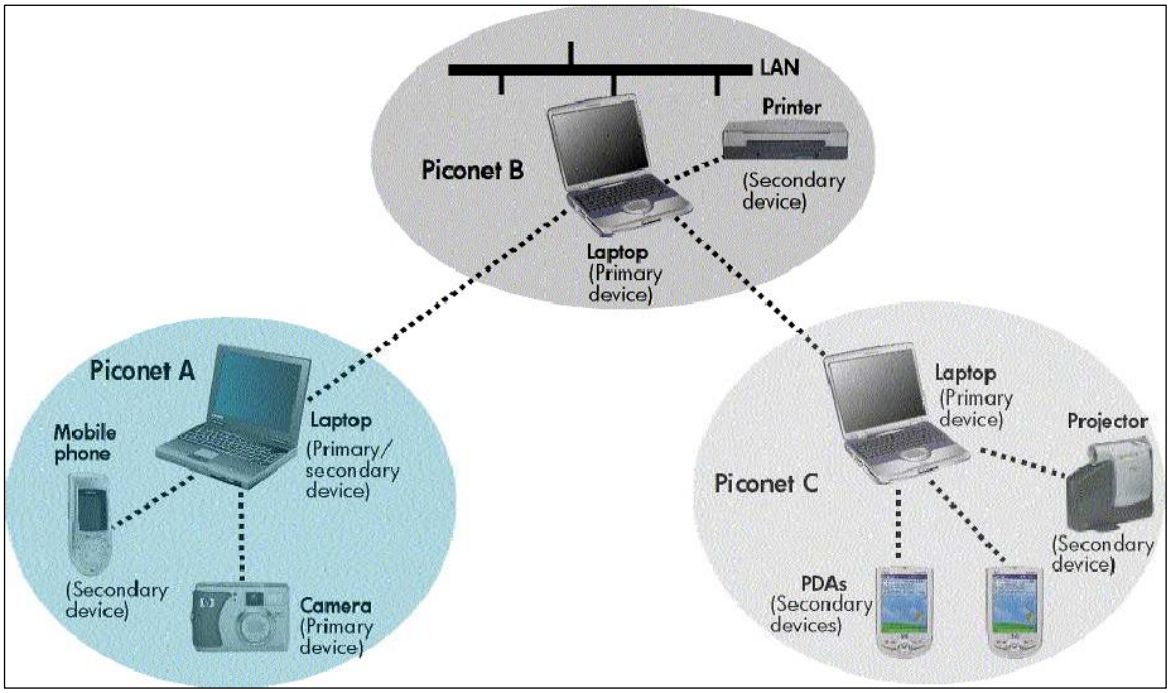


Fig2.6.1. A Scatternet

It is composed of interconnected piconets that maintain communication between more than 8 devices. Scatternets are formed when a device of one piconet that can be master or slave choose to act as a slave in second another piconet. Scatternets provide higher throughput. Also in different piconets multiple-hop routing between devices is possible. That means at one time only single component can interface in one piconet so they hop from one to another relying upon the channel limit.

CHAPTER 3
BLUE TOOTH BASED SENSOR NETWORK

To produce different countless applications the developers are proving operability between different devices. Wireless sensor networks are example of such applications. It has various small devices consisting of sensing unit, microprocessor, power source and wireless communication bond. Important features of wireless sensor networks: During the task in process network nodes combines with each other. It has a particular attention on data.

As the position of junctions is not decided in the field and formation of smart sensor nodes is not planned it could be experienced that some sensor nodes close in such positions that they either can accomplish needed measurement or the error probability is more [2] [4]. To overcome these problems a small node of repetitive number is disposed. These nodes additionally merge and share data. And thus ensures better outcomes. The collected data is send to the users by “gateway” using multiple hop routes in the Smart sensor nodes that are scattered.

3.1 FUTURE OF BLUETOOTH

- **BROADCAST CHANNELS:**

  Adoption of Bluetooth into mobile phones and enable advertising models based on users pulling information from the information points.

- **TOPOLOGY MANAGEMENT:**

  Automatic configuration ofpiconet topologies in scatternet situations.

- **QoS IMPROVEMENTS:**

  Enable audio and video data transmission at higher quality, especially in best effort traffic being transmitted in the same piconet.

3.3 BWT SECURITY

BWT security is complex, transparent, and easily implemented BWT uses 3 security mechanisms: *authentication*, *authorization* and *encryption*.

❖ **levels (modes) security are:**

MODE 1: No Security;

Anyone can use the device; default setting conditionin printers, etc.

MODE 2: Service Level Security;

permission required to access the device; exchange of business cards; personal authentication.

MODE 3: Link Level Security;

devices to be paired before connection and transfer.

3.3 BLUETOOTH HARDWARE ARCHITECTURE

Bluetooth hardware has 3 prime function modules:

2.4 GHz Bluetooth radio frequency Trans receiver unit. Link controlling unit Host controller interface Host controller is made up of a digital signal processing section with link controller and central processor. Link controller composed of both hardware and software parts for the execution of base band processing and physical layer protocols. CPU core helps Bluetooth module to filter page request and to handle enquiries. Link manager is software which runs on the CPU and communicates to them with the help of link manager protocols.

3.4 HOST CONTROLLER –

Consists of a *Digital Signal Processing* part, having *Link Controller (LC) & CPU Core.* It interfaces to the Host environment.

3.5 LINK CONTROLLER –

Consists of *Hardware & Software parts* to perform Base−Band Processing, and Physical Layer Protocols. Also perform slow−level digital signalprocessing to form connections.

3.6 CPU CORE –

Helps Bluetooth Module to handle Inquires andfilter page request (not involving host device).

3.7 LINK MANAGER –

LM software runs on CPU core. LM discovers other remote LMs and communicates. Them via LMP (link manager protocols). Bluetooth Module also incorporates Higher−Level Software Protocols, governing the functionality and interoperability with other modules.

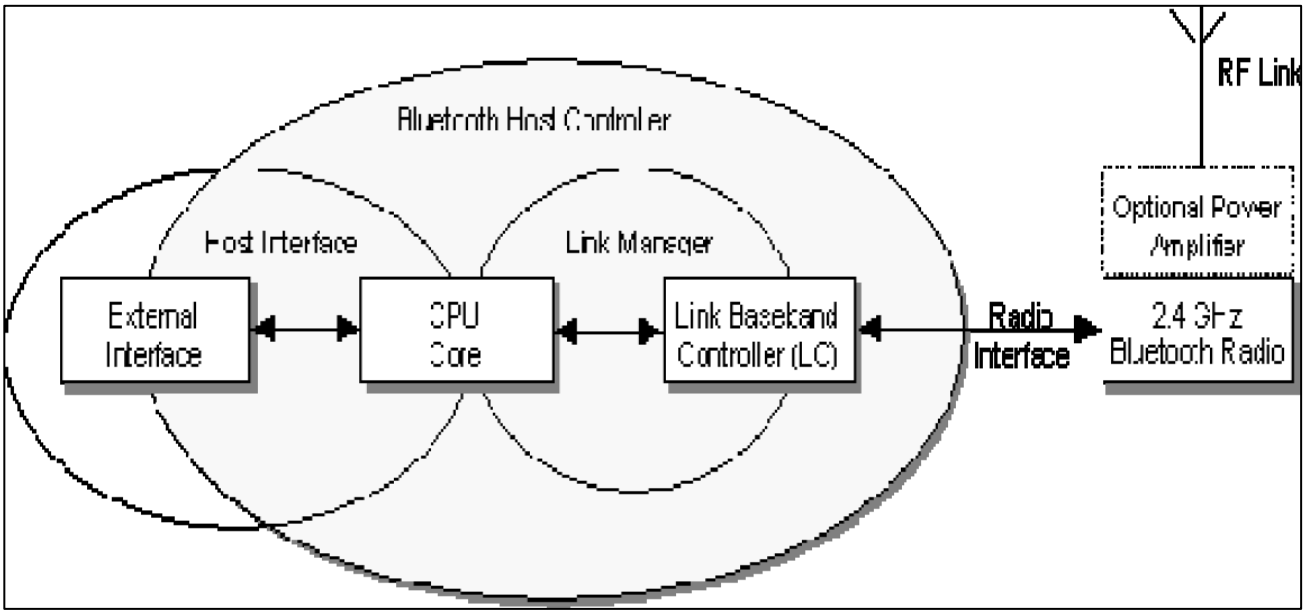


Figure 3.7: Bluetooth Architecture

Host controller is made up of a digital signal processing section with link controller and central processor. Link controller composed of both hardware and software parts for the execution of base band processing and physicallayer protocols. CPU core helps Bluetooth module to filter page request and to handle enquiries .Link manager is software which runs on the CPU and communicates to them with the help of link manager protocols.

3.6 MULTIHOP NETWORK ASSEMBLY

Because Bluetooth is connection oriented, the BTnodes need to be assembled into a multihop network before any .

3.7 Design

We must take into account the following characteristics of our platform when designing a multihop network assembly procedure:

Bluetooth Connection Establishment. Bluetooth connections are established between a master and a slave. The assembly procedure must establish the role of each node with respect to a connection. Note that nodes cannot exchange information before they have established a connection. In addition, slaves cannot communication with other slaves or overhear the communication taking place on other connections. As a result, we cannot use protocols involving spontaneous communication among neighbor nodes.

Dual Radio Approach. There are three possible con- figurations for each dual-radio node: (i) a node can be connected as slave on its two radios, (ii) a node can be connected as slave on one radio and as master with up to seven connections on the other radio, or (iii) a node can be connected as master with up to seven connections on both its radios**.**

Device Discovery Protocol. In order for two devices to discover each other, they must be in two complementary states at the same time: Inquiry and inquiry scan. The inquiring device continuously sends out is anybody out their messages hoping that these messages (known as ID packets) will collide with a device performing an inquiry scan. To conserve power a device

wanting to be discovered usually enters inquiry scan periodically and only for a short time known as the inquiry window. During this period, the device listens for inquiry messages.

3.8 The main challenges for the assembly procedure are thus

A second approach consists in configuring each node a priori. Each node is configured with a radio operating as a master and the other operating as a slave. This obliterates the need for the discovery and information exchange phases from the first approach. The second approach constitutes a baseline. We chose to implement it on top of our Tiny Bluetooth stack.

Our baseline solution, inspired by BlueTree (discussed below), is the following. When a node boots up, it enables one of its radios (the slave radio) and starts looking for an- other node to connect to. In this stage, the node will not be discoverable/visible for other nodes; it considers itself an orphan looking for a network. If it discovers other nodes, it tries to connect to one of them as a slave4. If the connection succeeds, it will consider itself member of the network, and turn on its other radio (the master radio), making it discov- erable and ready to accept connections from nodes that are not currently members of the network.

If the connection as a slave fails, it is because the master has reached its limit on the number of connections it can accept (recall that a master can connect to seven slaves). The node then tries to connect to one of the other nodes it has found in its vicinity. If there is no such other nodes ready to accept a connection then the node tries to connect again to the first node it contacted. If a master connected to seven slaves receives three repeated connection requests from the same node N, then it disconnects one of its slaves and accepts the connection from the node N. It has been shown that when a master is connected to more than five slaves, additional slaves are in connection range with at least one of the connected slaves. As a consequence, it is probable that the disconnected node will find a node that it can connect to in its vicinity**.**

In order to bootstrap the assembly procedure, one of the nodes, say a gateway, starts-up with its master radio turned on, thus being discoverable for other nodes. Its slave remains disabled, or is used to connect to the external world. The network topology obtained with this assembly procedure is a tree, that we call a connection tree, rooted at the gateway node that starts-up the assembly procedure. Note that if we

assume that all nodes start-up with their slave radio enabled, then the connection tree will be bushy: once a node activates its master radio, all nodes in its vicinity will have a chance to connect to it before they get a chance to connect to its children.

It is desirable that a sensor network be accessible via several gateways. One has a privileged role as the root of the connection tree. Other gateways are connected as leaves If the root of the connection tree fails, then all the nodes disconnect and the assembly procedure can restart with another gateway taking the responsibility of being root of the connection tree.

CHAPTER 4
WORKING

Sohrabi et al. describe two assembly procedures for the Sensoria dual-radio nodes (equipped with proprietary frequency hopping radios). The first procedure proceeds as follows: the two radios of a node are tuned to two fixed channels. During network assembly, messages are exchanged on those channels to form clusters (in each cluster a node is elected as a master while the other nodes are slaves). The constraint that a node cannot be master on its two radios ensures that the clusters constructed for the two radios are different. Such overlapping clusters cover the entire network with a high probability; but there is no control over the topology of the connection tree. The second procedure relies on a discovery phase during which one of the radios is tuned to a control channel in order to broadcast information. Once a node has received information about its neighbors it takes a decision on its own role: a master remains tuned to the control channel and exchange data to its slaves on its second radio while a slave exchanges data on both its radios. Both procedures rely on tuning one or both radios onto a fixed channel so that nodes can broadcast information to their neighbors. This is not possible with Bluetooth.

Basagni et al. [20, 2, 13] proposed a set of scatternet formation protocols for Bluetooth devices: Blue Tree, BlueStar, Blue Mesh. The basic protocol is Blue Tree that constructs a multihop network with a tree topology. Blue Tree proceeds in two phases. First, every node obtains information from its neighbors (nodes spend enough time inquiring and responding to discover their neighbors). Second, a designated node, the blue root, initiates the construction of the Blue Tree. This node is assigned the role of a master and its neighbors becomes its slaves. Recursively, each node that has become a slave is assigned the role of master with respect to its un- connected neighbors. The blue root is thus the root of the connection tree, all the intermediate nodes in the connection tree are both slaves and masters, while the leaves are just slaves. The constraint that one master cannot be connected to more than seven slaves is handled via tree reorganization. This work rely on the assumption that Bluetooth supports scatternets (intermediate nodes are master and slaves on a single radio). Our assembly procedure is an adaption

of the Blue Tree protocol for the dual-radio configuration. The main difference is that our approach does not include a discovery phase, rather nodes can join the network at any time (an unconnected node has its slave radio turned on and is looking for a master).

4.1 IN-NETWORK QUERY PROCESSING :-

Tiny DB developed at UC Berkeley and Intel Research [11, 12], processes declarative queries over sensor data within the sensor network. The proposed query language allows user to collect, filter and aggregate data produced in the sensor net- work. In-network query processing proceeds in two phases. First, the query is disseminated from a gateway node to all appropriate sensor nodes inside the network. This first phase results in the establishment of a routing tree rooted at the gateway and spanning all the sensor nodes producing data for the given query. Second, data is processed when transmitted up the routing tree.

Each sensor node runs an instance of TinyDB, which is responsible for (i) establishing and maintaining the rout- ing tree, (ii) processing query fragments over data received via the network or read on a local sensor and (iii) trans- mitting processed data up the routing tree. Aggregates are processed in a distributed manner: each node computes a partial state record based on the values it reads and obtains from its children. The partial state records are transmitted and the actual aggregate value is obtained at the root of the routing tree.

TinyDB is a TinyOS library. It relies on basic TinyOS building blocks essentially for sending and receiving data over the network, for setting the power saving mode, for setting timer events and for reading sensor data.

TinyDB has been designed for the Mica motes , assuming communication based on connection less broadcast on a shared channel radio. The choice of a Bluetooth radio challenges some of the assumptions that underly its design and implementation.

4.2 Topology Management :-

For each query, TinyDB constructs a routing tree rooted at the gateway on which the query is submitted. The rout- ing tree is constructed by having nodes pick one of their neighbors as their parent.

This procedure for routing tree construction is very much similar to the one we described in the previous section for network self-assembly. The construction of the BlueTinyDB routing tree consists in constructing a directed version of the connection tree, with the edges oriented towards the gateway on which the query is submitted. The oriented edges are maintained as a parent-child relationship (independent from the master-slave relationship maintained by the self- assembly component).

BlueTinyDB constructs the parent-child relationship while disseminating a query. Each node retrieves the list of neighbors from the self-assembly component. The node from which the query is received becomes the parent, the other neighbors become children, regardless of whether they are master or slaves. There are two issues to be considered:

There might be several gateways connected to a cluster of nodes and there might be several queries submitted to different gateways at the same time. Those queries will share the same connections. It is thus desirable that the connection tree is as bushy as possible, in order to minimize the height of the routing trees, regardless of the gateway to which a query is submitted.

Madden et al. [12] have proposed Semantic Routing Trees to restrict the distribution of a query to those nodes who actually participate in a given query. The idea is to maintain meta-data on each node to decide whether the nodes accessed through a given child will participate in the answer to the query (in which case the query is distributed further to this child) or not (in which case the query is not distributed to this child). The fact that all routing trees share the same connection tree facilitates the collection and the maintenance of this meta-data.

4.3 TDM at the Radio Level :-

TinyDB organizes a time division multiplexing (TDM) scheme at the application level in order to organize the processing and the transfer of data along the routing tree. The design of the TDM scheme described by Madden et al. is driven by the following observations:

The output of a TinyDB query is a stream of values. To each value is associated a time-stamp. Aggregates are performed on values whose time-stamp belong to a same time interval, or epoch. Because the processing of an aggregate is distributed across a set of nodes, it should be synchronized across nodes in order to ensure that data is indeed processed within an epoch. It is desirable to power down a node while it is not processing or transmitting data.

TinyDB's TDM scheme is implemented at the application level as follows. First, the query is disseminated together with timing information that allow nodes (i) to synchronize clocks and (ii) to set the timer that will drive the TDM scheme. Then each node is put to sleep (i.e., the microcontroller is forced into power saving mode using the TinyOS Snooze command) until it is woken up by a timer event. When a node is asleep, it cannot process sensor or network data. Once a node is awake it receives data from its children, reads sensor data, processes the query locally and transmit the partial state record to its parent. The timers are set on each node so that there is an overlap between the intervals children are sending and parents are listening. The main problem with this approach is the timer set- up: How to assign time slots in the TDM scheme at the application level? The duration of each slot depends on the duration of an epoch (defined in the query), the height of the tree (data produced during an epoch should be processed together), the breadth of the tree (each node must receive data from all its children before it transmits processed data up the routing tree) and the density of nodes (there can be collisions between nodes within transmission range whether they are siblings in the routing tree or not). The TDM scheme also relies on clock synchronization across nodes, which is only approximate at the application level5.

Using Bluetooth, it is possible to let the radio drive the TDM scheme. Bluetooth relies on a TDM scheme at the physical and MAC layers to implement the frequency hop- ping transmission. It also provides a power saving mode that allows reducing the duty cycles between connected devices: the sniff mode. In sniff mode, two devices negotiate specific slots where communication can begin. A mas- ter can enter the sniff mode for each individual connection it manages. The period of the sniff slots is a parameter given by an application when a connection enters the sniff mode. The application has however no control over the timing of the sniff slots

CHAPTER 5

Separated Channels

TinyDB relies on the TinyOS MAC layer to avoid collisions between nodes transmitting during the same interval. Those collisions occur not only between parents and children but between any two nodes who are in transmitting range of each other. If the density of nodes is high...then we can expect a high rate of collisions. As a consequence, TinyDB monitors channel contention and adaptively reduces the number of packets sent as contention rises.

The problem of channel contention is not acute in the context of a Bluetooth radio. Each connection constitutes a separated channel. There is fenly interference between pairs of nodes hopping on the sanie frequency at a given point in time. This is a marginal problem.

Note that separated channels are expected to boost the performance of flooding as collisions are avoided between parents and children as well as between branches of the rout ing tree. Note however that separated channels do not permit the snooping used in TinyDB to optimize the performance of certain aggregate operators (such as MAX) or to com pen sate for the loss of some messages.

5.1 Calibration of the Binodes :-

The Tiny Bluetooth stack allows us to run experiments with the BJ nodes, in order to calibrate them with respect to throughput as well as energy consumption.

5.2 Code Footprint :-

Our first challenge was to squeeze a tiny version of the Bluetooth stack within the Bloods. The code footprint for our Tiny Bluetooth stack is less than 3 KIB. It is thus comar cable to the native TinyOS stack (approximately 2 KIB [9]) and one order of magnitude less than the Smart-its Blue tooth stack (approximately 30 KiB).

In the dual radio configuration, we have duplicated the Tiny Bluetooth stack for the sake of simplicity. The code footprint of these stacks together with network assembly and multiboot, routing is approximately 8 KiB. We estimate that we can reduce the code footprint by 30 to 40% if we avoid duplicating the stack. The data footprint (taking into account the noninitialized variables-traditionally, denoted as bass) is 1 KIB due to the packets statically allocated in the Tiny Bluetooth stack.

Table 1 breaks down the code, bas (non initialized variables) and data (initialized variables) footprint for the dual radio configuration**.**

5.3 Throughput :-

Bluetooth specification promise high throughput. Can our Tiny Bluetooth stack take advantage of this potential on the Binodes?

We measured throughput on point-to-point connection between master and slave. Figure 5 shows the results we obtained for all possible Bluetooth encodings and two pay load sizes.

The distinction between initialized and non-initialized variables is interesting as non-initialized variables are not part .

Description	code	bss	data
Support & TinyOS core	1180	0	
UART0 & interrupts	346	4	
UART1 & interrupts	292	5	
hciPacket0	604	155	
hciPacket1	588	155	
hciCore0	1624	159	
hciCore1	1590	159	
Assembly component	4796	1021	16
Total	11020	1658	16

Table 1: Code size breakdown (bytes)

Figure 5.3.1

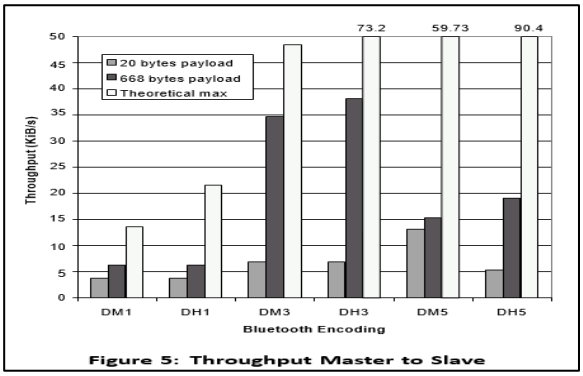


Figure 5: Throughput Master to Slave

Figure 5.3.2

Bluetooth defines 6 encoding that correspond to the com- bination of two levels of resistance to noise (due to different levels of redundancy in the encoding – DM: high resistance and DH: low resistance) and three configurations of the TDM scheme (the node transmits for 1, 3 or 5 time slots). The two payload sizes we consider are 20 bytes, i.e., enough to transport a few integers which is the case for many sensor network applications, and 668 bytes which is the maximum HCI packet size, i.e., enough to transport high bandwidth traffic such as images or sounds.

For payloads of 20 bytes, the encoding used does not make a significant difference. We achieve a throughput of about 5 KiB/sec. For the maximum payload of 668 bytes, through- put is significantly higher: it varies from 6 to 35 KiB/sec. Here, the number of slots used for transmission and the resistance to noise have a significant impact. Our result shows that the best throughput is achieved with 3 slots transmissions (more data is transmitted than with 1 slot transmission and there is less retransmissions than with 5 slots).

Note that the throughput we achieve is far from the theoretical max. First, our Tiny Bluetooth stack accesses the Bluetooth module via the UART and the maximum through- put supported by the UART is approximately 45 KiB/sec8. Second, the Bluetooth module generates superfluous mes- sages that take up bandwidth on the UART interface (hard- coded Ericsson string events).

5.4 Throughput for an multipoint asymmetric DM3 connection, aggregate and individual band- width :-

Additional experiment have shown that slave to master communication achieves a throughput which is very similar to the master to slave communication shown in Figure 5. This means that data can flow up the routing tree regardless of whether individual connections are master-slave or slave- master.

The master is responsible for allocating the channel band- width for each slave in a multipoint connection. We setup the master to connect to a number of slaves and send packets in round-robin order on each connection. We measured the bandwidth on each connection. It would be expected that the total bandwidth stays the same and that each slave receives a fair share.

Table 2 shows that the aggregate bandwidth drops considerably but that each slave receives a fair share of that bandwidth. Hence the ROK modules wastes in-air slots by not sending meaningful data.

Madden et al. [write that it is reasonable to expect that the Mica motes transmit approximately 500 bytes per second. We thus achieve with the BTnodes a throughput, which is between one and two orders of magnitude higher for point-to-point connections and at the very least a factor of two for multipoint connections.

CHAPTER 6

ENERGY CONSUMPTION

The throughput results are encouraging; however, energy consumption is the key metric for the calibration of the BTnodes. We thus measured both current draw and voltage9 for different regimes of our experimentation platform. Figure 6 summarizes our results.

Our first goal was to measure energy consumption for an idle BTnode (in black on the figure). According to the manufacturer , the 8 MHz MCU can use up to 12 mA at 5 V (60 mW) in idle sleep mode. With a slightly lower clock frequency, we observe a lower energy consumption (about 8 Note that we had to modify TinyOS to operate the UART at full speed. The UART relies on two registers for sending bytes: one, called the shift register, contains the byte currently being sent, the other contains the byte to send next. We fixed the UART module so that it generates an event each time the UART moves a byte to the shift register in- stead of generating an event each time the shift register is empty and the UART has no more data to send.

9We used an input voltage of approximately 5V. We observed that voltage varied slightly during the experiments. We thus decided not to assume constant voltage to compute the energy consumption.

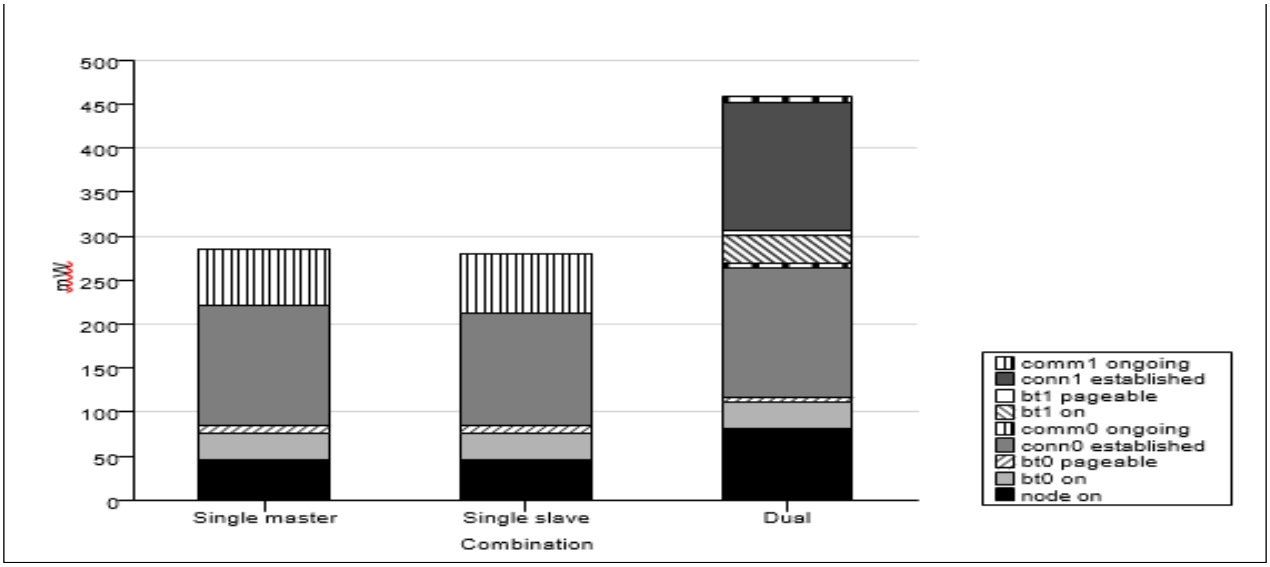


Figure 6.1

This is mostly due to the (unused but turned on) led on the BT tester, which consumes about 22 mW, as well as the voltage regulator and the serial voltage level converter. Turning a Bluetooth radio on consumes about 30 mW extra (in idle mode). Making the radio pageable, or in quarriable requires an additional 9 mW. A BTnode with a single radio waiting for a connection thus consumes 89 mW. In the dual radio case, turning on both radios and making the pageable and in quirable consumes about 155 mW. Nodes use about 136 mW to maintain connections. Additional experiments showed that putting a connection in sniff mode saves a marginal 5 mW. This suggests that some optimizations might have been missed in the Bluetooth module design.

Once a connection is established sending or receiving data consumes an additional 65 mW when transferring at 6 KiB/- sec when using a single radio and 5 mW for each radio when transferring 10 packets per second.

All in all, a single-radio BTnode uses approximately 50 mW when idle and 285 mW when communicating, while these numbers are up to 80 and 450 mW for a dual-radio BTnode. If we compare with the Mica motes from UC Berkeley, these numbers are high. Indeed, Madden et al. report 10 mW for for an idle node and 60 mW when communicating.

The BTnodes consume five times more energy than the Mica motes doing nothing! This is due to the fact that the MCUs are placed in different sleep modes. As we have seen with TinyDB in Section 5.1, the Mica motes favors applications that manage themselves the time they spend in sleep mode. This way, the MCU can be put to sleep in power save mode, where only the external clock can send wake-up signals, i.e., the motes do not get data from sensors or from other nodes while the MCU is in sleep mode.

6.1 The *power save* sleep mode is the most energy efficient :-

The BTnodes favor applications that are in sleep mode until events are received from the Bluetooth radio or the sensors. This precludes the use of the power save mode because events generated by the Bluetooth radio would be ignored. The MCU is best put in idle sleep mode until an event is received from the UART or from the clock. The idle sleep mode is however less energy efficient than the power save mode. We can estimate to 8 days the life expectancy of a BTnode in idle sleep mode with two standard AA batteries. We noted that energy consumption was high when trans- mitting data. However, if we consider the ratio energy per bit transmitted, then the BTnode exhibits rather good performances Running at 6 KiB/s, we obtain approximately 5.5 µJ/bit for a single radio and 10 µJ/bit for two radios. These numbers are close to the ones reported in the original TinyOS paper (approximately 4 µJ/bit) for an ideal transmission without interference or loss. According to Madden et µJ/bit

Another characteristic of these results is that connection maintenance consumes a lot of energy, both on master and slave. Using the sniff mode does not make a significant difference. These results suggest that connections should only be established for short periods. There is thus a trade-off between the cost of connection maintenance and the cost of network assembly. We explore this trade-off in the next Section.

In summary, all these remarks suggest that the BTnodes are well suited for applications that are active over a limited time period, with few unpredictable bursts of very heavy network traffic (taking advantage of the high throughput). An example of such application could be a sensor network deployed to secure a building in a mounted operation in urban terrain. Such a network could have a life expectancy of up to a week, operating in sleep mode until individuals are detected in which case as much situational information as possible could be obtained (including possibly images or sound on a suite of point-to-point connections).

6.2 Network Self-Assembly :-

Because it is expensive to maintain a connection it is likely that the network will be assembled repeatedly when data needs be transmitted. Network assembly should thus be rapid and energy efficient.

Figure 7 shows energy consumption as a function of time on a BTnode during network assembly. The figure is annotated with letters corresponding to the different phases of the experiment. As the node is turned on (a), it initializes its *slave* radio. After a while, it discovers its parent-to-be, connects to it (b) and enables its *master* radio that becomes discoverable. The connection is established within 20 seconds.

For the sake of clarity, we turn on the children nodes one after the other. After 30 seconds, the first child is turned on, it is detected and a connection is established on the node's *master* radio (c). After 40 second, the second child is turned on and a second connection is established (d). Energy consumption corresponds to the calibration presented in the previous Section. Additional connections result in a very limited increase in energy consumption (about 3-4 mW).

Once those connections are established, the node routes data from both children to its parent: both children send packets with payload of 5 and 50 bytes every 10 seconds for a 100 seconds. Note that the children do not send in synch. The peaks of energy consumption we observe from 40 seconds until 160 seconds correspond primarily to the master .

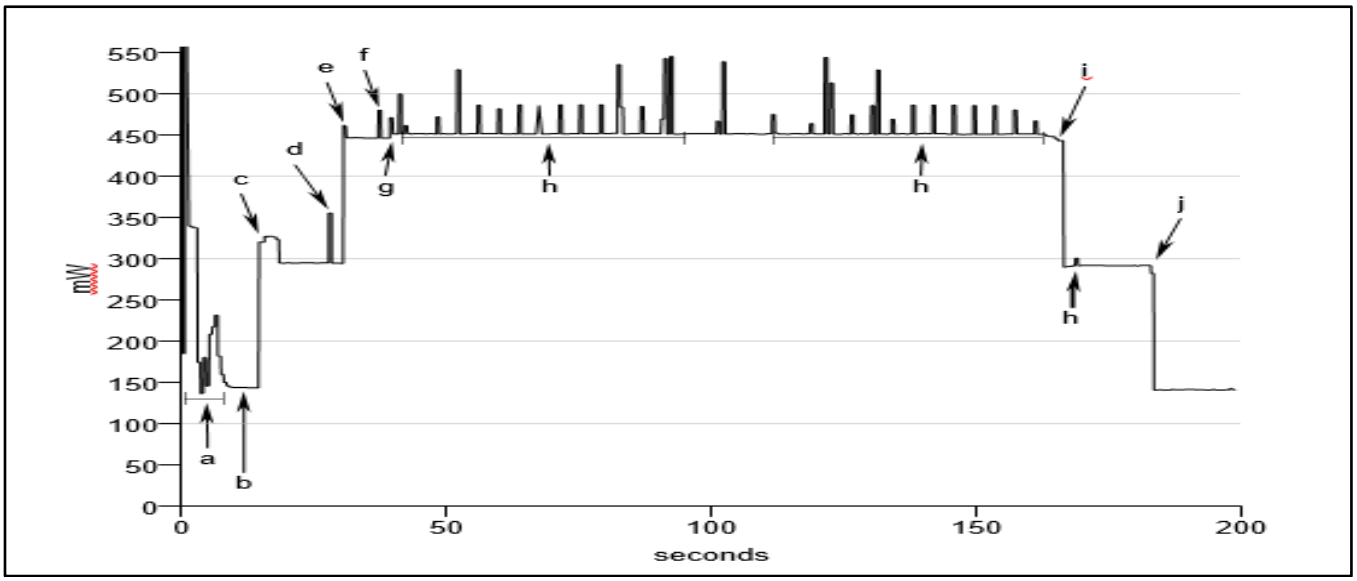


Figure 6.2

6.3 Energy Usage during Network Assembly :-

Radio being discoverable (e) and also to the transmission of data packets (f)10. After 160 seconds, both children are disconnected (g) and after 180 seconds the parent is disconnected (h). The node remains idle with both Bluetooth radios disabled but powered on.

Additional experiments show that our assembly procedure requires in most cases between 5 and 10 seconds per level in the connection tree assuming all nodes have already initialized their Bluetooth radios11. Indeed, children can only be discovered once the connection to the parent is established. Energy consumption corresponds to what could be predicted after calibrating the nodes. There is no extra over- head introduced by the assembly procedure. In order to save energy, it thus seems a good idea to assemble the network repeatedly, for relatively short periods during which high volumes of data can be transmitted.

6.4 TOOH BASED SENSOR NETWORKS

- **Challenge:** It is to ensure interoperability among various Bluetooth manufactures' devices and to provide numerous applications.
- **One such application is** : WIRELESS SENSOR NETWORKS (WSN)
- **Important features of WSN:** Collaboration of network nodes during execution and Data Centric nature. Many smart sensor nodes scattered in the field collect data and send it to users via 'gateway' using multi−hop routes.
- **WIRELESS SENSOR NETWORKS (WSN):** WSN consists of number of small devices equipped with a sensing unit, microprocessors, wireless communication interface and power source**.**

Two main operations performed by WSN are:

- **QUERING** − Queries are used when user requires only the current value of the observation.
- **TASKING** − More Complex operation Used when a phenomenon has to be observedover a large period of time.

6.5 **Functions of GATEWAY :-**

- **Communication with sensor networks** : Shortage Wireless Communication ; Discovery of smart sensor nodes
- **Gateway Logic*:*** Controlling Gateway interface and data flow ; Providing uniform access to sensor
- **Communication with users*:*** Communication over Internet,WAN, Satellite, etc.

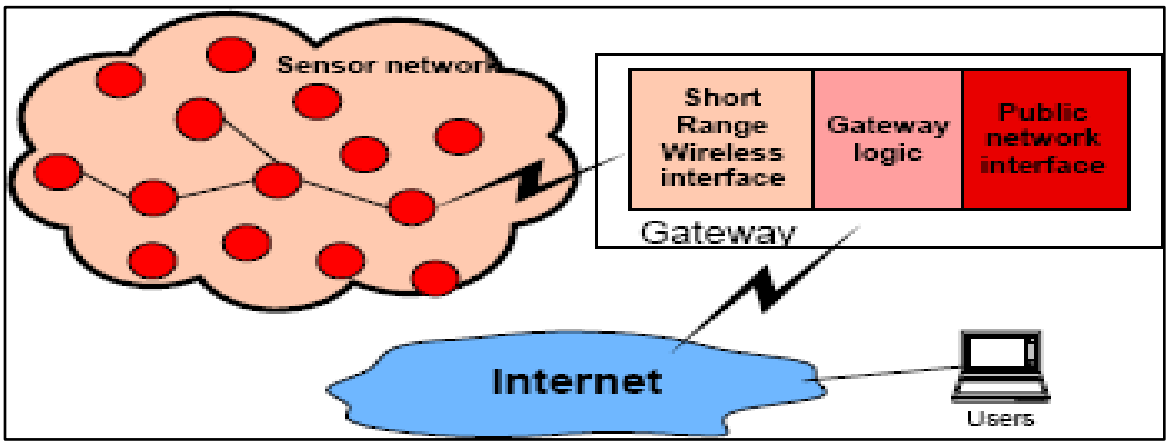


Figure 6.5 : **Wireless Sensor Network**

CHAPTER 7

ALGORITHM FOR OPERATION OF NETWORKS

- Initialization of gateway and Bluetooth Inquiry Procedure.
- Discovery of Bluetooth device and Checking of major and minordevices.
- Setting of parameters and assigning type of devices and sensors.
- Description by Service−Class Field.
- Discarding of non−smart nodes.
- Else, service database of the discovered smart node is searched forsensor services.
- If no current sensor profile, then database is searched for serial portconnection parameters.
- Lastly, Bluetooth link is established and data exchange with smartnode starts.

7.1 SENSOR NETWORK IMPLEMENTATION

OBJECTIVE: To build a Hardware plat form and genericSoftware Solutions to serve for research in WSN protocols.

Components of Sensor Network: **Smart Sensor Nodes** and **Gateway**

Gateway and Smart nodes are members of *piconets* and so,not more than 7 nodes can exist in the network.

Example: Pressure Sensor For implementation of Pressure Sensor as Bluetooth Node,following components are **important:** Bluetooth Device Sensors Microcontroller

TEDS − Transducer Electronic Data Sheet

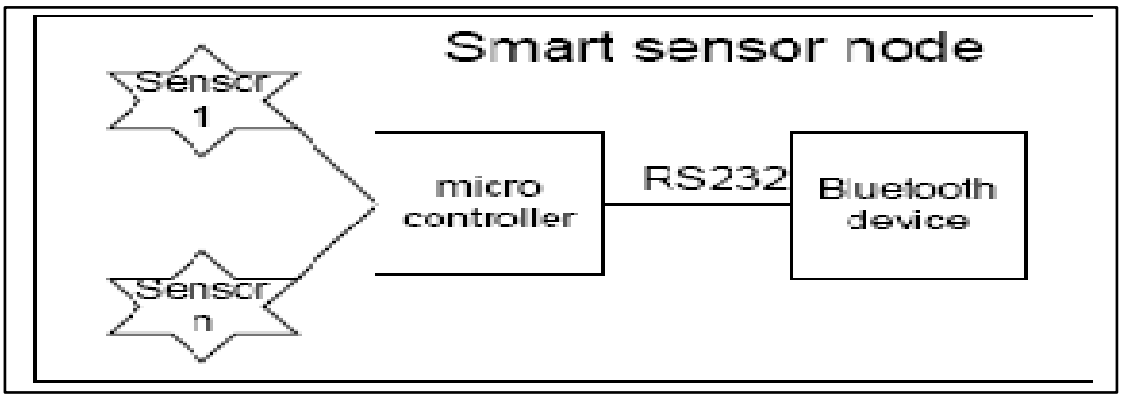


Figure 7.1 : SMART SENSOR NODE IMPLEMENTATION

7.2 Wireless connectivity over Bluetooth:

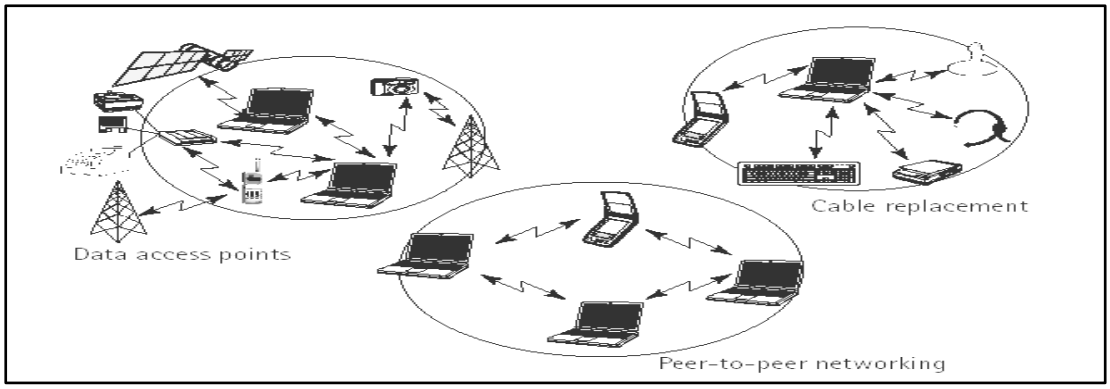


FIGURE 7.2

The basic function is to provide a standard wireless technology to replace the multitude of propriety cables currently linking computer devices. Better than the image of the spaghetti-free computer system is the ability of the radio technology to the network when away from traditional networking structures such as business intranet. for, example imagine being on a business trip with a laptop and a phone. The Bluetooth technology allows interfacing the two. Then picture meeting a client and transferring files without cabling or worrying about protocols. That is what the Bluetooth will do.

7.3 Network arrangement:

Bluetooth network arrangements can be either point-to-point or point-to-multipoint. The various network arrangements regarding Bluetooth are:

a) Single-slave

b) Multi-slave(up to 7 ‘Slaves’ on one master)

c) Scatternet

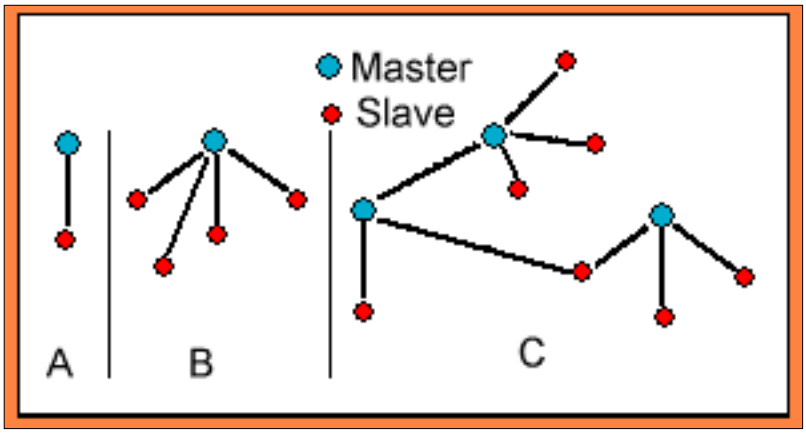


FIGURE 7.3

7.4 Bluetooth:

Bluetooth has been designed to operate in a noisy radio frequency environment, and uses a fast acknowledgement and frequency-hopping scheme to make link robust, communication wise. Bluetooth radio modules avoid interference from other signals by hopping to a frequency after transmitting or receiving a packet. Thus, Bluetooth modules use Frequency-Hopping Spread Spectrum (FHSS) techniques for voice and data transmission. Compared with other systems operating in the same frequency band, Bluetooth radio typically hops faster and uses shorter packets.

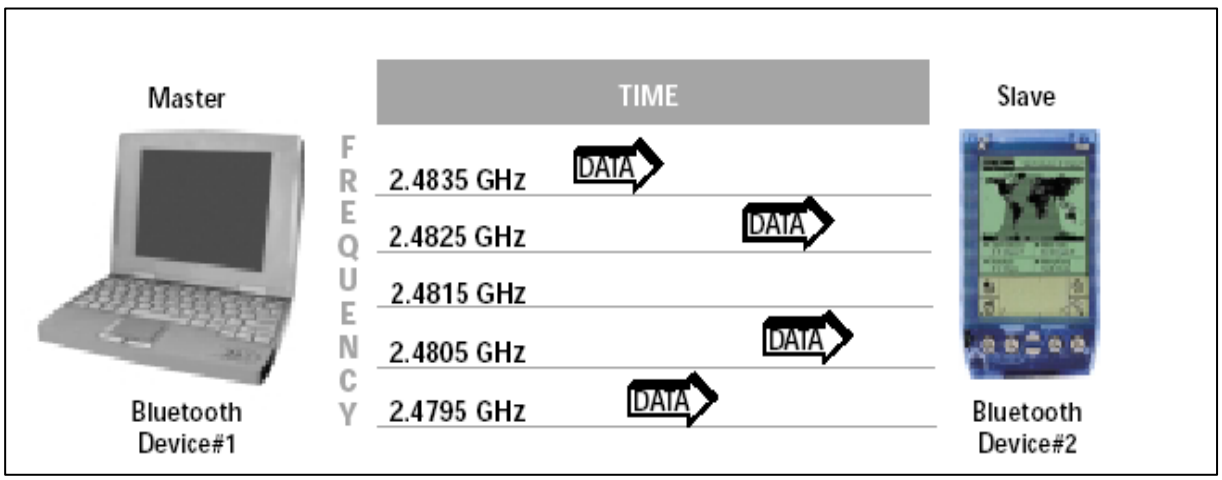


FIGURE 7.4

FHSS uses packet-switching to send data from the transmitter of one Bluetooth module to the receiver of another. Unlike circuit-switching which establishes a communication link on a certain frequency (channel), FHSS breaks the data down into small packets and transfers it on a wide range of frequencies across the available frequency band. Bluetooth transceivers switch or "hop" among 79 hop frequencies in the 2.4 GHz band at a rate of 1,600 frequency hops per second.

7.5 Bluetooth – "Simply the Best!":-

Bluetooth competes with existing technologies like IrDA, WLAN and Home RF. IrDA is not omni directional and it is a line-of-sight technology. WLAN's are essentially ordinary LAN-protocols modulated on carrier waves, while Bluetooth is more complex. Home RF is a voice and data home networking which operates at a low speed. Bluetooth hops very fast (1600 hops/second) between frequencies, which does not allow for long data blocks. These problems are overcome in Bluetooth and that's why Bluetooth is **considered** "Simply the Best".

7.6 BLUETOOTH BASED SENSOR NETWORK:-

The main challenge in front of Bluetooth developers now is to prove interoperability between different manufactures' devices and to provide numerous interesting applications. One of such applications is a wireless sensor network.

Wireless sensor networks comprise number of small devices equipped with a sensing unit, microprocessors, and wireless communication interface and power source. An important feature of wireless sensor networks is collaboration of network nodes during the task execution .Another specific characteristics of wireless sensor network is Data-centric nature.

As deployment of smart sensor nodes is not planned in advance and positions of nodes in the field are not determined, it could happen that some sensor nodes end in such positions that they either cannot perform required measurement or the error probability is high. For that a redundant number of smart nodes is deployed in this field. These nodes then communicate, collaborate and share data, thus ensuring better results.

Smart sensor nodes scattered in the field, collect data and send it to users via "gateway" using multiple hop routes.

7.7 A Wireless sensor network

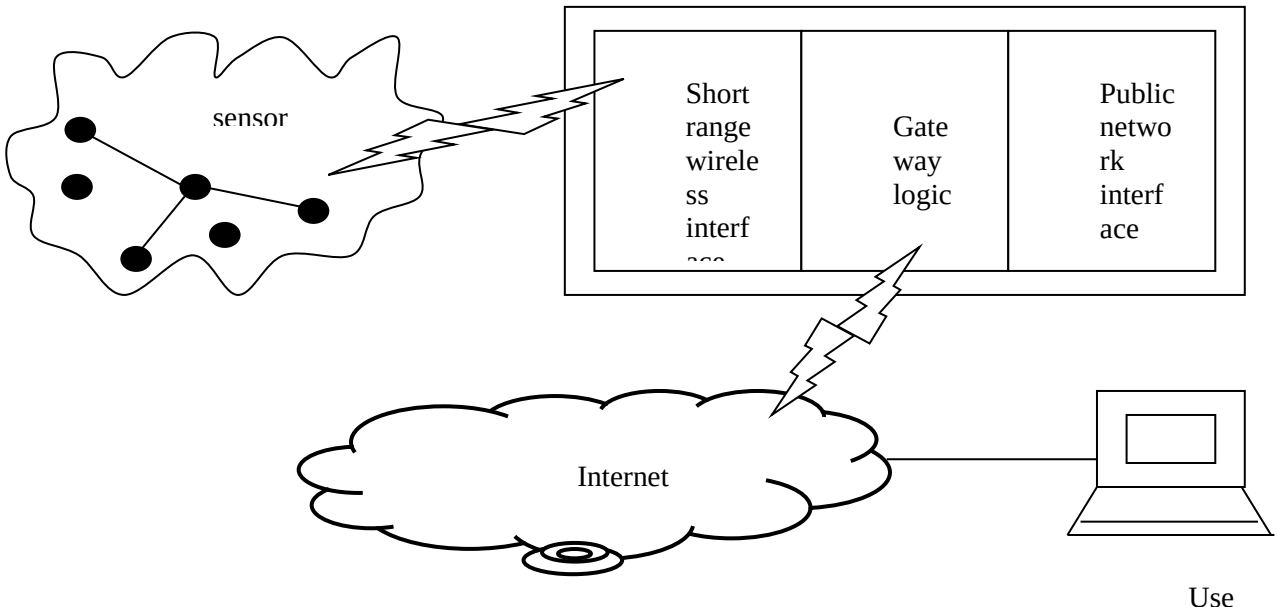


FIGURE 7.7

The main functions of a gateway are Communication with sensor Networks

- Shortage wireless communication is used.
- It provides functions like discovery of smart sensor nodes, generic methods of sending and receiving data to and from sensors, routing.

❖ Gateway logic

- It controls gateway interfaces and data flow to and from sensor network.
- It provides an abstraction level that describes the existing sensors and their characteristics.
- It provides functions for uniform access to sensors regardless of their type, location or N/W topology, inject queries and tasks and collect replies.

❖ Communication With Users

Gateway communications with users or other sensor networks over the Internet, WAN, Satellite or some shortage communication technology. From the user point of view, querying and tasking are two main services provided by wireless sensor networks. Queries are used when user requires only the current value of the observed phenomenon. Tasking is a more complex operation and is used when a phenomenon has to be observed over a large period of time. Both queries and tasks of time to the network by the gateway, which also collects, replies and forwards them to users.

CHAPTER 8

SENSOR NETWORK IMPLIMENTATION

The main goal of the implementation was to build a hardware platform and generic software solutions that can serve as the basis and a test bed for the research of wireless sensor network protocols.

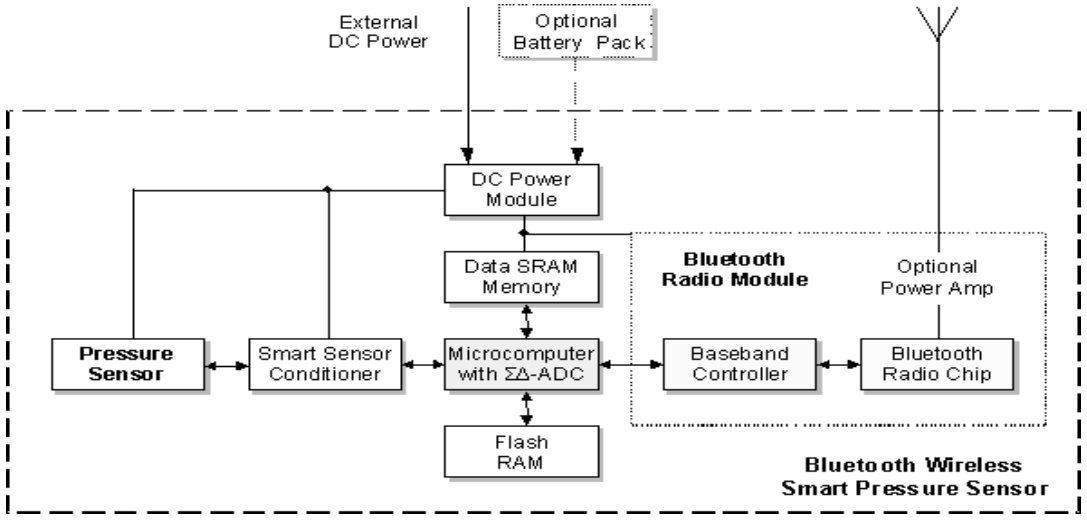
Implemented sensor network consists of several smart sensor nodes and a gateway. Each smart node can have several sensors and is equipped with a micro-controller and a Bluetooth radio module.

Gate way and smart nodes are members of the Piconet and hence maximum seven smart nodes can exist simultaneously in the network. For example, a pressure sensor is implemented, as Bluetooth node in a following way.

The sensor is connected to the Bluetooth node and consists of the pressure-sensing element, smart signal-conditioning circuitry including calibration and temperature compensation, and the Transducer Electronic Data Sheet (TEDS). These features are built directly into the sensor microcontroller used for node communication control plus memory for TEDS configuration information.

8.1 Smart Sensor Node Architecture

The architecture shown in figure can easily be developed for specific sensor configurations such as thermocouples, strain gauges, and other sensor technologies and can include sensor signal conditioning as well as communications functions.



A Bluetooth wireless smart pressure sensor node

FIGURE 8.1

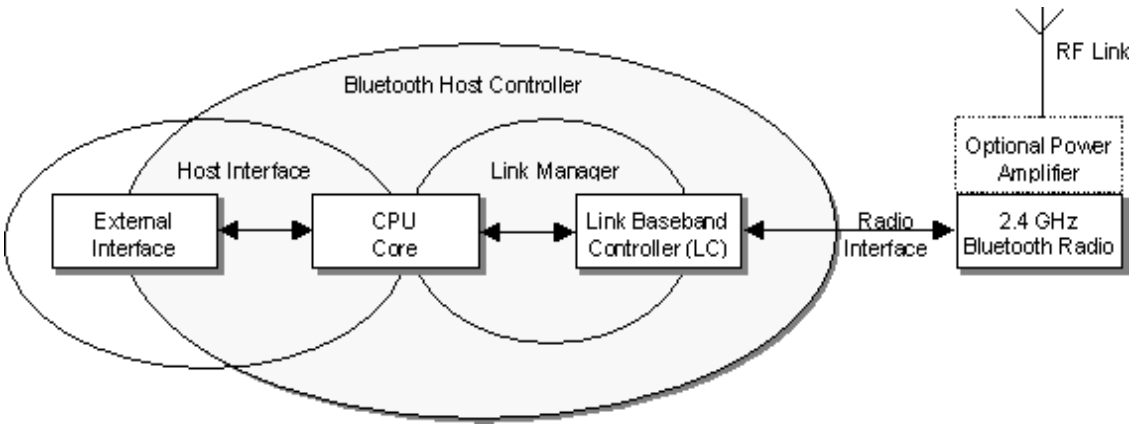
Conditioned along sensor signal is digitized and digital data is then processed using stored TEDS data. The pressure sensor node collects data from multiple sensors and transmits the data via bluetooth wireless communications in the 2.4 GHZ base band to a network hub or other internet appliance such as a computer.

The node can supply excitation to each sensor, or external sensor power can be supplied. Up to eight channels are available on each node for analog inputs as well as digital output. The sensor signal is digitized with 16-bit A/D resolution for transmission along with the TEDS for each sensor. This allows each channel to identify itself to the host system. The node can operate from either an external power supply or an attached battery. The maximum transmission distance is 10 meters with an optional capability to 100 meters.

The IEEE 1451 family of standards are used for definition of functional boundaries and interfaces that are necessary to enable smart transducer to be easily connected to a variety of networks. The standards define the protocol and functions that give the transducer interchangeability in networked system, with this information a host microcomputer recognized a pressure sensor, a temperature sensor, or another sensor type along with the measurement range and scaling information based on the information contained in the TEDS data.

A blue tooth module consists primarily of three functional blocks – an analog 2.4 GHz., Blue tooth RF transceiver unit, and a support unit for link management and host controller interface functions.

The host controller has a hardware digital signal processing part- the Link Controller (LC), a CPU core, and it interfaces to the host environment. The link controller consists of hardware and software parts that perform blue tooth based band processing, and physical layer protocols. The link controller performs low-level digital-signal processing to establish connections, assemble or disassemble, packets, control frequency hopping, correct errors and encrypt data.



FUGURE 8.1

8.2 Bluetooth module Hardware Architecture :-

The CPU core allows the blue tooth module to handle inquiries and filter page request without involving the host device. The host controller can be programmed to answer certain page messages and authenticate remote links. The link manager(LM) software runs on the CPU core. The LM discovers other remote LMs and communicates with them via the link manager protocol (LMP) to perform its service provider role using the services of the underlying LC. The link manager is a software function that uses the services of the link controller to perform link setup, authentication, link configuration, and other protocols. Depending on the implementation, the link controller and link manager functions may not reside in the same processor.

Another function component is of course, the antenna, which may be integrated on the PCB or come as a standalone item. A fully implemented blue tooth module also incorporates higher-level software protocols, which govern the functionality and interoperability with other modules. Gate way plays the role of the Piconet's master in the sensor network. It controls establishments of the network, gathers information about the existing smart sensor nodes and sensor attached to them and provides access to them.

8.3 Discovery Of The Smart Sensor Nodes :-

Smart sensor node discovery is the first procedure that is executed upon the gateway installation. It goals to discover all sensor nodes in the area and to build a list of sensor's characteristics and network topology. Afterwards, it is executed periodically to facilitate addition of new or removal of the existing sensors. The following algorithm is proposed.

When the gateway is initialized, it performs Bluetooth inquiry procedure. When the blue tooth device is discovered, the major and minor device classes are checked. These parameters are set by each smart node to define type of the device and type of the attached sensors. Service class field can be used to give some additional description of offered services. if discovered device is not smart node it is discarded. Otherwise, service database of the discovered smart node is searched for sensor services. As currently there is no specific sensor profile, then database is searched for the serial port profile connection parameters. Once connection strings is obtained from the device. Blue tooth link is established and data exchange with smart mode can start.

8.4 A WIRELESS SENSOR NETWORK:-

Wireless sensor network provides two important actions querying and tasking.

Querying: When there is need of current value of observed event, queries are used. It is more composite operation and is useful when an event needs to be noticed for a long time.

- These two services querying and tasking are allocated to the system through "gateway" which also forwards thecollected responds to users.
- The main functions of a gateway are
- Interaction with sensor networks.
- Short wireless transmission is used.
- It contributes functions like finding of smart sensor nodes
- Provides techniques of sending and receiving data from sensors, routing.
- It controls gateway attachment and dataflow to and from sensor network.
- It provides standard of dealing with ideas that gives detail about the current participating sensors andtheir characteristics.
- It allocates function for consistent approach to sensors without being affected by their type, location or network topology, introduce queries and tasks and gather respond.
- Communication with Users.
- Gateway communication with user and another sensor networks through the internet, WAN, Satellite orother short communication technology.

8.5 SENSOR NETWORK IMPLEMENTATION:-

The main target of the sensor network execution was to construct a hardware plan and common software answers that can works as support and for the motive of research in wireless sensor network agreement.

Implemented sensor networks involve smart sensor nodes and a gateway. Every node has countless sensors and is connected with microcontroller and Bluetooth element. Gateway and smart nodes behaves as the part of the Piconet. For example: A Bluetooth pressure sensor.

The sensor is attached on the Bluetooth node. It has the pressure feeling element, smart signal-conditioning circuitry containing density and temperature reception, and the Transducer Electronic Data Sheet (TEDS). These are integral features of sensor microcontroller which are used for node communication control and also for the memory for TED's arrangement.

8.6 SMART SENSOR NODES DISCOVERY :-

For the execution for the gateway installation the first step is the locating of the smart sensor nodes. With the help of gateway, all the smart nodes are found and build the list of the attribute of sensors and network topologies. After that for making the process easy for the removal of existing sensor and addition of the new it is executed concurrent .

8.7 ALGORITHM :

After the initialization of gateway is done, Bluetooth performs the inquiry procedure. Major and minor device classes are checked after the detection of the Bluetooth devices. Some parameters are set by the sensor nodes for defining the type of device and type of attached sensor. To give some further explanation of offered service, the duty class field is used. And if the founded device is not a smart node then it is disposed. As there is no particular sensor profile at ongoing time, the database is hunt for the serial port outline link. Once interrelated string is acquired then the Bluetooth link is accepted and the transfer of the data is started.

CHAPTER 9
ADVANTAGES

9.1 ADVANTAGES :-

- Inexpensive.
- Low Power utilization.
- Small range.
- Wireless Technology.
- Sensible throughput.
- Cheap maintenance cost.
- Easy link formation.
- Share voice and data.

9.2 LIMITATIONS :-

- Low data range.
- Interference with other device.
- Low security.
- Low data rate.

9.3 ACKNOWLEDGEMENTS :-

We thank Oliver Kasten and Jan Beutel from ETH Zurich for interesting discussions and for their help with the BTN- odes. We also wish to thank Janus B. Lundager for his time and his multi-meter. We appreciate the feedback from the anonymous reviewers and from our shepherd Mark Smith

9.4 APPLICATIONS :-

1. Constant sensing
2. Health monitoring
3. Event detection & local control of actuators
4. Smart buildings
5. Smart grid and energy control system

CHAPTER 10
Conclusion

10.1 CONCLUSION:-

Wireless sensor networks are fascinating research area with multi feasible applications and with many solutions. They are combination of various tiny devices having the ability of interacting and dealing with data. Bluetooth is an easy and suitable option for data communication in sensor networks. To plan routing and application-level procedures, we overlook multiple affairs related to MAC layer, physical layer, application layer and routing layer. For the automatic link up and information exchange, the Bluetooth devices need to bring within the range of another device.

Our experiments with the BTnodes suggest that Blue- tooth based sensor nodes could be appropriate for a niche of applications exchanging unpredictable bursts of data during a limited time period. It is difficult to predict that Bluetooth will emerge as an alternative of choice for a larger class of sensor network ap- plications. First, as long as scatternets are not supported we will have to rely on a dual-radio approach which is ex- pensive and energy inefficient.

Second, using Bluetooth for commercial purpose requires a certification (to guarantee that heterogeneous devices can communicate) which is irrelevant and unrealistic in the context of sensor networks. Third, the encapsulation of the three lower layers on hard- 10A similar experiment without data transmission also exhibits energy peaks but with a more regular intensity.

11We observe that connections take most of the time 5 to 10 seconds with some outliers requiring 20 to 30 seconds. We have studied the characteristics of device discovery and connection establishment in previous work.

CHAPTER 11
References

11.1 REFERENCES:-

[1] Mark Weiser, “The Computer for the Twenty-First Century”, Scientific American, 1991.

[2] Wireless World Research Forum, “Book of Visions”, http://www.wireless-world-research.org

[3] Scenarios for Ambient Intelligence, EU IST Advisory Group

[4] H. Gharavi and K. Ban, “Video-based multihop ad-hoc sensor network design,” in Proc.WorldWireless Congress, SanFrancisco, CA, May 2002, pp. 469–474.

[5] J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust", ACM/IEEE Intl. Conf. on MobileComputing and Networking (MobiCom 99), Seattle, WA, August 17-19, 1999.

[6] G.J. Pottie, W.J. Kaiser, “Wireless Integrated Network Sensors”, Communications of the ACM, May 2000, Vol. 43, No.5

[7] J.M. Rabaey, M.J. Ammer, J.L. da Silva Jr., D. Patel, S. Roundy, “PicoRadio Supports Ad Hoc Ultra-Low PowerWireless Networking", IEEE Computer Magazine, July 2000

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, K. Pister, “System Architecture Directions for Networked Sensors”,Proceedings of the ASPLOS 2000