AN IMPLEMENTATION OF CONSENSUS THROUGH BLUETOOTH COMMUNICATION

Yinan Wang

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

May 2014

APPROVED:

Shengli Fu, Major Professor and Interim Chair of the Department of Electrical Engineering
Xinrong Li, Committee Member
Yan Wan, Committee Member
Costas Tsatsoulis, Dean of the College of Engineering
Mark Wardell, Dean of the Toulouse Graduate School

Wang, Yinan. *An Implementation of Consensus through Bluetooth Communication.* Master of Science (Electrical Engineering), May 2014, 56 pages, 14 figures, references, 20 titles.

This thesis provides an implementation of consensus of multi-agent networked systems. Consensus problem is an important issue of distributed computing and has various algorithms and applications in the field of electronical and computer science. The consensus requests all nodes of a network reach an agreement over a certain measurement. An algorithm of convergent consensus problem is implemented through a small network of Bluetooth communication in the thesis. The connections of the Bluetooth devices are wireless, and the device nodes of the network are driven by C++ software and Winsock API. The simulation results show that the implementation completes all the requirements of the distributed consensus algorithm.

ACKNOWLEDGEMENT

I would like to express sincere gratitude to my advisor Shengli Fu who offers me requisite devices and tools for the thesis. His attitude to research inspired me. This thesis would not have been possible without the help and encouraging.

I would like to thank all professors who have given me lectures.

Finally, my sincere thanks also go to all people who have helped me.

LIST OF CONTENTS

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

Consensus problem is an important issue of distributed computing and has various algorithms and applications in the field of electronical and computer science. Examples of applications of consensus include message diffusion, statistics, and group unit identification. A maximum-likelihood estimate can be computed by using averaging algorithm, in the case that nodes are linear in variables and noise is Gaussian [1].

For a network, the consensus problem requires that all nodes reach an agreement over a certain measurement. Consider a graph with many nodes and edges, the agreement is a certain quantity to be achieved. Agents (nodes) can interexchange information with his neighbors via the edge. This procedure is controlled by a consensus algorithm which is a rule to specify how information interchanges between agents of a network [2]. For a small graph or for few nodes, the convergence rate is quick and the consensus algorithm is relatively simple. But for large networks, convergence rate is indeterminate or even not convergent, and the consensus algorithm is more complicated [3], [4], [5].

1.1 Implementation

In the literature, there exists a lot of research on the consensus problem, but most of researchers discuss the synoptic performance from theoretical perspectives, for example, the convergent behavior and convergence rate [3], [4]. In this thesis, I study an implementation over personal digital assistant (PDA) to obtain the performance of real-world sensors.

Mobile devices are widely used in daily life. Typical techniques for information

exchange include WiFi and Bluetooth. Bluetooth mode communications set up between two or several units, and any unit can quit at any moment without declaration. This enables a non-centralized communication, which is one reason for choosing Bluetooth for this thesis [6].

Consider a network composed by several sensors, and each device is independent and equal. Each needs to detect some environmental data; thereby every sensor has a value, such as temperature of the area. The sensors connect through Bluetooth and their network is a typically an undirected graph. Therefore, wireless channels and devices are the edges and nodes, respectively, in a graph. Through messages exchange, the value of each agent is expected to reach the average of the initial values.

In this article, the process of exchanging data via Bluetooth will be discussed. Theoretically, the algorithm over three PDAs as a small network is implemented. Data and computation procedure are simultaneously recorded in devices.

1.2 Development Platform and Tools

1.2.1 Tools

Microsoft Visual Studio 2008

Windows mobile SDK

C and C++

Bluetooth

1.2.2 Devices

The devices are the Dell personal digital assistant of Axim family of Windows Mobile

powered Pocket PC Devices. The devices used in this thesis were produced in 2006.

1.2.3 Winsock

Winsock is an API (Application Programming Interface) for developing Windows programs, so that applications can deliver and receive messages via the TCP/IP protocol. Winsock has been standardized by BSD UNIX socket. The windows operating system provides the socket API, so that application programs can use the network [18].

In the program of this thesis, Winsock forms the crucial part of the communication and creates an access port for connection. The development environment is Microsoft Visual Studio 2008 and Microsoft Windows Software Development Kit by C++. The wireless communications go through Bluetooth.

Similar to other development tools, Winsock has many components in its core definition files. For example, Sockets Address is a symbol which includes device addresses and ports.

Using this address, internet sockets recognize and deliver data to a target device. Chapter 4 will discuss some function components such as “SOCKET” “Connect” “Send” and ”Recv”.

1.3 Outline

This thesis is organized as follows. Chapter 2 provides basic informations of implementation and concepts of Bluetooth and discusses Winsock, platforms, and tools. Chapter 3 describes the consensus problems, algorithms and the applications. Chapter 4

presents the Bluetooth implementation of the average consensus algorithm with numerical result. Chapter 5 concludes the thesis.

CHAPTER 2

BLUETOOTH SYSTEM

Bluetooth is a technology standard which has been developed for radio links in small range. The advantages of Bluetooth include low power, simple structure and low cost, which make it pretty suited to mobile devices. To communicate with other devices as simply as possible, Bluetooth can replace the cables and connect to the portable electronic devices wirelessly. As an IEEE standard, any Bluetooth device regardless of manufacturer can connect to any other Bluetooth device in their working range. Bluetooth devices connect and communicate wirelessly; these devices form a network called a piconet. Each device has simultaneously connection with up to seven other units per piconet (this may increase in new Bluetooth version). In addition, one device node can simultaneously belong to different piconets. These piconets are dynamical and change working state automatically when Bluetooth devices enter and leave the radio range [7].

2.1 Bluetooth History

Bluetooth was developed by Ericsson of Sweden in 1994 [7]. Its name comes from Harald Blaatand "Bluetooth" II, king of Denmark from 940 A.D to 981A.D.

To develop a specification for short range wireless connection, the Bluetooth Special Interest Group (SIG) was founded by Intel, IBM, Toshiba, Ericsson, and Nokia in February 1998. Now the group is also supported by Microsoft, 3COM, and Motorola. About 2500 companies have joined SIG [7].

2.2 Bluetooth System Requirements

Bluetooth standards consider the design of wireless connection to be universal, so that there are many applications for the Bluetooth. The wired connection should be abandoned in Bluetooth standards and the standards offer several of short range services to create more opportunities for new usages. Bluetooth system is not only considered as a cable replacement technology, but also opens new areas of wireless communications.

The most important requirement for the wireless link is a common framework which can directly transmit information across different devices in efficient method. Users expect to interact with server device wirelessly; therefore Bluetooth must offer services for cooperation of all devices where each device provides its function and interface. Ad hoc connections must be enabled to establish in the standards [8].

Another feature is unconscious connectivity, which means devices can connect to others almost without any users' interaction or operation, Bluetooth can also support voice and data transmission [8].

One major concern of wireless communication is information secrecy. It is not allowed to compromise of security in replacing wired links. While the Bluetooth requires no registration of new services with central authority, communications should be protected in similar methods as in cable networks [8].

2.3 Protocol Stack

The main Bluetooth protocol stacks include:

Logical Link Control and Adaptation Protocol (L2CAP) are defined by Bluetooth SIG.

L2CAP is core protocol of Bluetooth and parallel works with Link Manager Protocol (LMP). It depends on the device to device baseband link which is provided by Bluetooth hardware. L2CAP layer is one part of Btd.dll [9].

Service Discovery Protocol (SDP) is a protocol to discovery Bluetooth services. It runs discovery service before connecting. SDP client module runs Btdrt.dll and SDP server module runs Btd.dll [9].

Link Manager Protocol (LMP) is the protocol to control linking establishment between Bluetooth devices. The LMP is an important part of GMPLS (Generalized Multiprotocol Label Switching) protocol stack, the latest LMP protocol specification is established by IETF (The Internet Engineering Task Force) in October 2003. It mainly realizes the function of optimal network link management and includes authentication and encryption [10].

RFCOMM (Serial Cable Emulation Protocol) is one part of the TS07.10 protocol. It derives point to point protocols and offers the communication path between different applications of different devices [9].

Transport Driver Interface (TDI): in Windows Embedded CE operating system, it connects Winsock and lower layer protocols.

COM Port Emulation: it enables to create virtual COM ports over RFCOMM channels in Windows Embedded CE. It controls LAN access profiles [9].

Personal Area Network (PAN) defines the protocol to support standard IP network service. It is also called the piconet, and it can include both wire and wireless devices.

Human Interface Device (HID): it supports human input devices, Such as keyboard and mouse [9].

Host Controller Interface (HCI) is an interface to Bluetooth hardware. It is responsible for linking establishment and controller management [10].

Bluetooth Universal Transport Manager (BthUniv): it detects the Plug and Play (PnP) device. It is an intermediate transport driver which loads the appropriate transport driver when PnP devices are plugged in. Bluetooth Universal Transport Manager has been built in Bthuniv.dll file.

Except the HCI transport, each layer works as an independent unit.

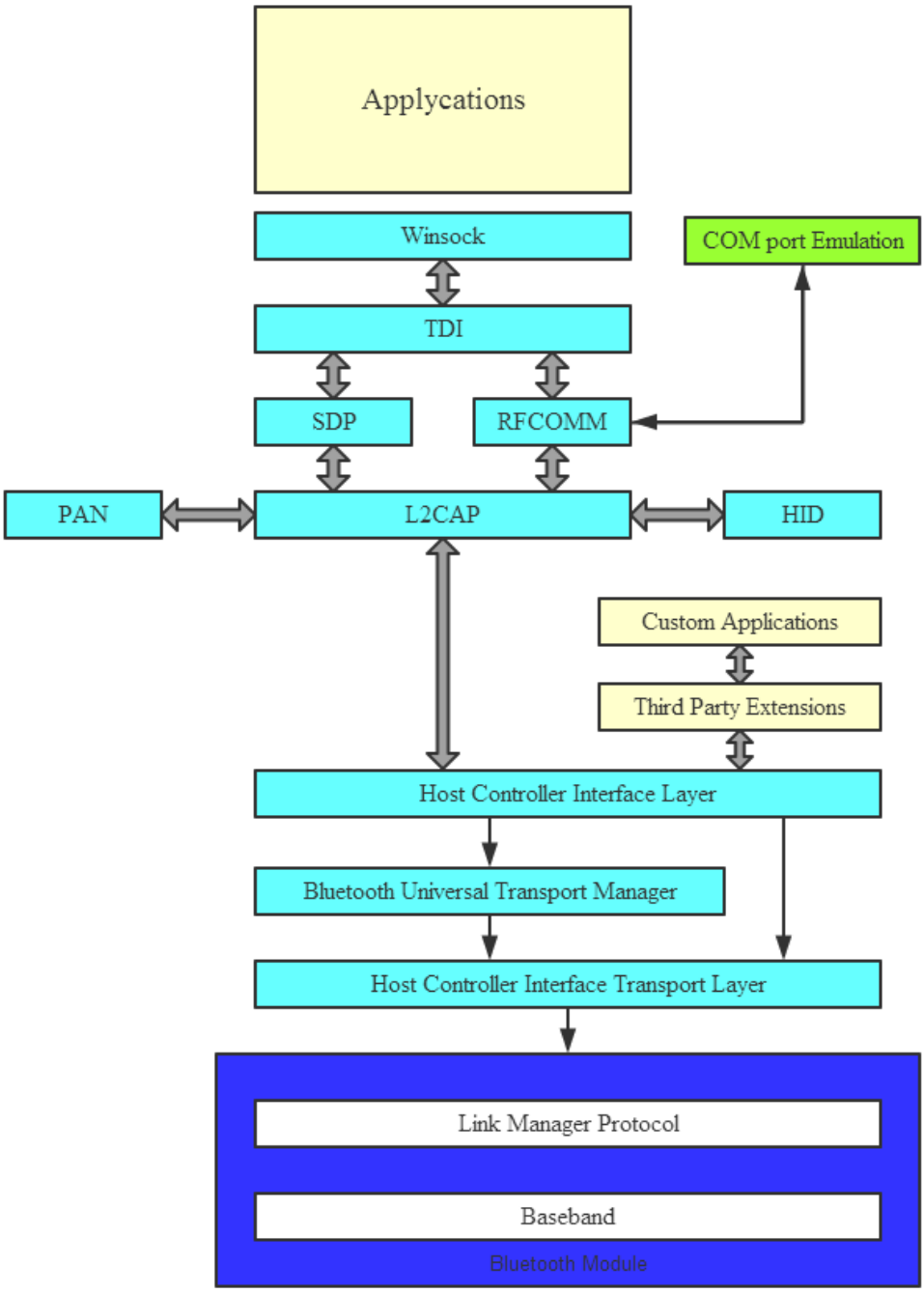

Figure 2.1: Bluetooth Protocol Stack [11]

Figure 2.1 shows the Bluetooth protocol stack. The baseband and LMP (Link Manager Protocol) are at bottom. They establish and control links between Bluetooth devices. Typically these two layers are implemented in firmware or hardware. The Host Controller layer interfaces the Bluetooth hardware to L2CAP and it may not be required, if L2CAP is

already on Bluetooth module. Then the L2CAP can communicate with LMP and baseband. In this way, protocol stacks perform layer by layer and on the top of the figure is the application. The application asserts claims, and then protocol stacks drive hardware to offer service to application [11].

2.4 Radio Layer

Radio layer works in ISM band around 2.4GHz that called free license. The band covers from 2400 to 2483.5 MHz in most of countries and the whole range is used to optimize the spectrum spreading [12]. Users can use this band freely if the radio power is under specified power and keeps noninterference to other devices.

Different from the other wireless technologies, frequency hopping (FH) technique is used for spread spectrum in Bluetooth systems. Since multiple uncoordinated networks may operate in the same band, fast FH and short data packets have great advantage to avoid interference. For example the error rate should be very high if a microwave oven operates at this frequency, because microwave oven and many other devices can operate the same frequency around 2.4GHz. FH and short data packets are efficient to reduce the interference. In addition, continuously variable slope delta modulation (CVSD) encoding can also reduce the bit error rates while message transmitting. Also redundant error correction scheme protects the packet headers to be identified correctly [13].

The frequency hopping are specified at 2402+k MHz (k= 0, 1..., 78). Nominally hop rate is 1600 hops every second, so that a single hop slot is 625 microseconds [13]. Message transmission relies on slot is odd or even [14]. The modulation is Gaussian FSK which uses

carrier frequency changing to transfer digital information and it is a frequency modulation. The transmitter power is 0dBm to 20dBm when the range is increased from 10m to 100m [15].

2.5 Bluetooth Packet Structure

Bluetooth packet includes an access code, header, and a payload. Access code is 68 or 72 bits, header is 54 bit and payload is from 0 to 2745 bits. The access code can detect different packets. The header includes address, link, and error control. The payload carries data. In the header, flow flag indicates whether the payload buffer is full or not and HEC (Header Error Check) is for CRC (Cyclic Redundancy Check) check. ARQN and SEQN check packet acknowledgment for the flow control. There are two basic types of data packets for the payload: ACL and SCO. ACL is asynchronous connection and SCO is synchronous connection [15]. Bluetooth packet format is shown in figure 2.2.

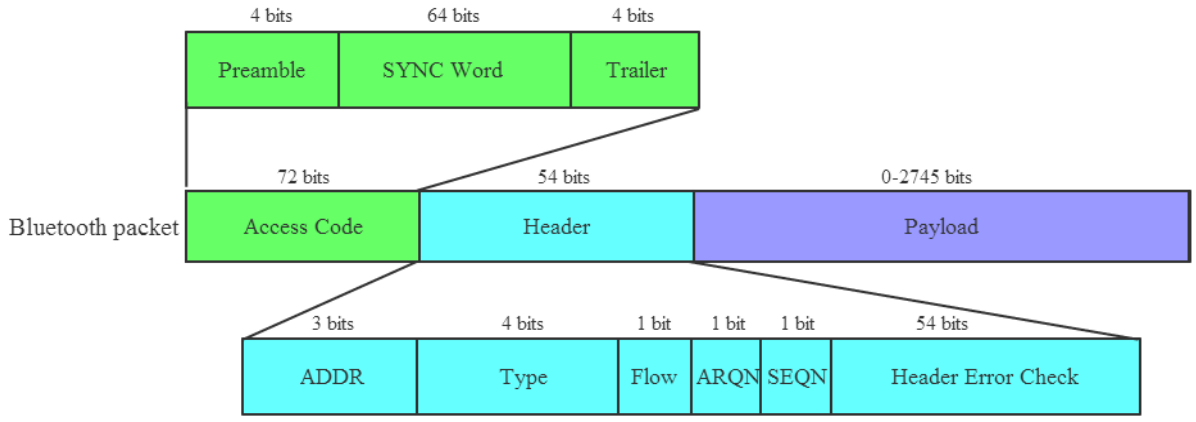

Figure 2.2: Bluetooth Packet [15]

2.6 Data Capacity

Theoretical data rate of Bluetooth is 1Mbps. However the available data rate is 723

kbps, because the header needs to leave space for informing different Bluetooth protocols. Bluetooth supports an asynchronous data channel or three simultaneous synchronous voice channels. It can also carry a channel which supports synchronous voice and asynchronous data at same time [7].

One voice channel carries 64 kb/s synchronous voice. A channel can support about 723.2 kb/s for asynchronous link, or 433.9 kb/s for symmetric link [7]. In fact the error correction capability decides the real data rate, because it determines what type of packet is used [15].

2.7 Relationship of Units

A piconet is composed of a group of devices which are connected with common channel. Usually the first device which initials the connection is regarded as master. The master manages the channel. Any two devices can communicate over SCO or ACL links if they are within one piconet. Any devices must enter a piconet, and then it can communicate with the connected devices. One device may not belong to only one piconet; it can be a unit of many piconets simultaneously. A master can be a slave in another piconet and similarly a slave can be a master in another piconet [13]. However, only one master is allowed in a piconet and up to seven active slaves within it. Figure 2.3 shows the relationship of devices in piconet.

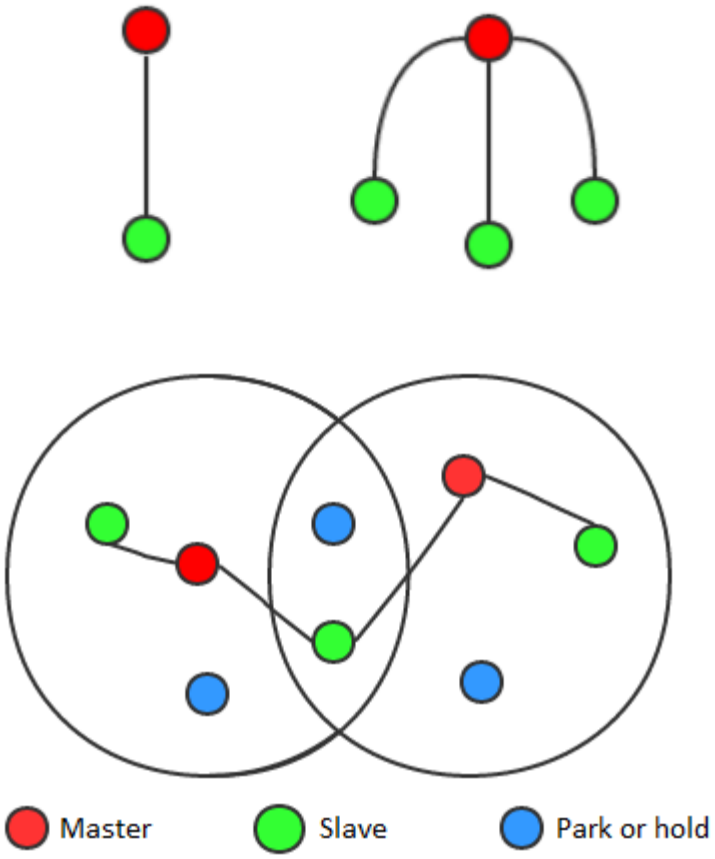

Figure 2.3: Piconets and Scatternet [11], [12]

2.8 Application Layer

Applications may directly access the L2CAP with the support of some protocols such as RFCOMM, TCS and SDP, and then the applications can use TCP/IP protocols. They are all allowed in Bluetooth standards. Applications may use SDP to discover service if a remote device is available. Point to Point protocol (PPP), File Transfer protocol (FTP) or other specific protocols are also allowed if required by the applications [7].

2.9 Software, Firmware and Hardware

Figure 2.4 shows the basic structure of Software, firmware and hardware [11].

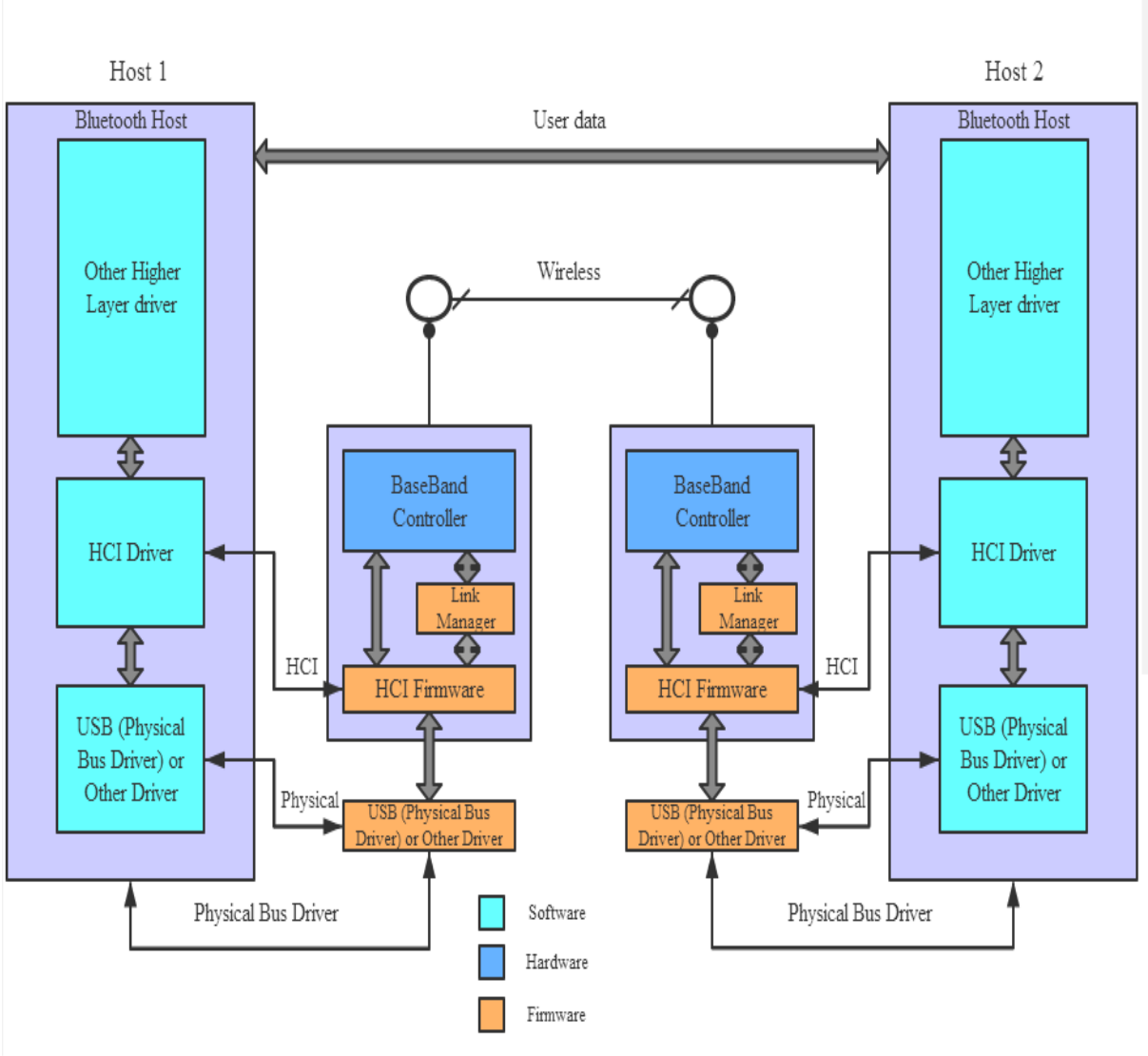

Figure 2.4: Structure of Software, Firmware and Hardware [11]

2.10 Connection

If a Bluetooth device wants to connect to another device, it should follow several procedures.

First step is inquiry. When the device starts the Bluetooth program, it searches the environment. If it finds access points in its working range, it will send inquiry information to other devices. This will result in two events, all nearby connectable devices respond message with their addresses and the device picks one from the responding devices [7], [9].

Then second is page. It is a procedure prior to connection. This procedure synchronizes the device with the access point. The synchronization informations which including clock offset, phase, and required initializations will be sent to other devices [7], [9].

Now link is able to establish. The type of link depends on the application. ACL link will be used in asynchronous connections such as receive email or pictures. SCO is used for synchronous applications [7], [9].

The above steps react on the process of connection. Service Discovery Protocol (SDP) is under control of LMP to discover what services are available. Different services are offered to different level authorities.

Device will create an L2CAP channel for the access point which has been detected by SDP. Application may use it directly. Depending on the application needs, RFCOMM will be created over L2CAP channel. By using RFCOOM, applications can create serial ports to offer service over Bluetooth platforms [10].

If the master of a piconet restricts its access point, a PIN code is needed. After the discover procedure, a security request will be sent to the device which intends to join the piconet. This step is called “pairing.” The PIN is not transmitted through the wireless channel, thereby knowing correct PIN code can access the service, otherwise system will deny the connection. In complete and official Bluetooth software, service is offered after successful paring. Also encryption combines with secure mode to guarantee the connections are authorized. In this program, pairing and PIN code is not necessary, because the Bluetooth device address is known [13].

Point to Point Protocol (PPP) link is a data link protocol. It establishes direct connection between two units and provides connection authentication, message encryption, and data compression [17].

After completing the above protocol, the network protocols (TCP/IP) is setup to send and receive data.

2.11 Connection states

During the processes of connecting to other devices, Bluetooth defines the following states.

2.11.1 Inquiry

In this state, device will send an Inquiry packet to nearby devices with a particular Bluetooth address. The information is repeated at 16 frequencies which form a hop sequence. Once a device begins to inquiry, there are two hop sequences. Each hop sequence must be repeated 256 times to receive all responses if the environment is error free. Total time will be 10.24 seconds and may be less if responses are enough to know the nearby devices' addresses [13].

2.11.2 Inquiry Scan

In this state, nearby devices allow themselves to be discovered. It means devices will send responding informations to the inquiry device. The responding informations should be enough to cover the 16 frequencies. A device can enter inquiry state when it is at connected or standby states [13].

2.11.3 Inquiry Response

When a device receives an inquiry message in inquiry scan state, it will send a response packet which contains the responding device address. However, the device will not send the response message immediately, because if there are many devices, sending too many response messages simultaneously will result in a collision. So the responding device waits for a random slot then sends FHS (Frequency Hopping Synchronization) packet to inquiry device. The FHS packet contains device address and clock information, then the device enters page scan state [13].

2.11.4 Page

The master device uses clock information in this state and other device might be listening in page scan mode. From the inquiry response information, the master device calculates access code of the other devices, and then it sends a page message. Note that it does not know whether the other devices have entered page scan state or not. Because of this, the page will be repeated several times, unless at least one response is received. Total time of collecting responses will take about 2.56 seconds [13].

2.11.5 Page Scan

Connection state or standby state can enter page scan state. The other devices are listening to page packet address in different mode and frequency [13].

2.11.6 Page Response

Once the slave device receives page message, it will respond the page device. It sends back messages containing its ID packet at the frequency which the page messages were

received. If the master device receives the page response, it will send its FHS packet to slave. Then the slave device sends its ID packet again, so that the page device can determine the channel access code of the piconet. If the device refuses to connect, it will return a NULL packet. If the response is successful, page response state is end. Both two devices will enter the connected state. Otherwise, page fails and the device returns error informations [13].

2.11.7 Active, Hold and Park Mode

In active mode, the connecting units can participate in a piconet. Message is transmitted in alternating slots. The master uses all even number slots and slave uses subsequent slots. Normally, the procedure of transmitting is synchronized.

In hold mode, ACL link to other device is hold. The slave does not support ACL packet in this mode. However, it keeps device information; if necessary it can come out of hold mode. In this mode devices can do inquiring, scanning, paging, or connecting to other device.

When devices stay in Park mode, there are very few actives which result in very low power consumption. The message from master device will be sent to broadcast channel, and park device regularly listens for the beacon signal and keeps synchronization [13].

2.11.8 SDP and L2CAP in Connecting Step

SDP is used to discover available services. A Bluetooth device will also run a SDP server if its services want to be discovered. SDP server maintains a record of services which the device allows to offer. In SDP, a device first pages slave device. If the slave device is scanning for this, it will respond the page. If device decides to access the service, it needs to

start a L2CAP connection. Then a connection to sever on the other device will be established with the informations obtaining from SDP. SDP is only for showing service, and accessing through a protocol which is based on L2CAP.

L2CAP only supports ACL and is based on the channels. L2CAP link is identified by channel identifiers, such as TCP/IP. L2CAP is set up after ACL link. It has low overhead with no CRC error check, because it relies on baseband for security and ordered delivery. L2CAP receive message from lower or higher layers, and offers support for service [15].

CHAPTER 3

CONSENSUS PROBLEM AND ALGORITHM

Similar to [3], the network of multiple agents system can be represented by a graph $G = (V, E)$. V is the set of nodes of graph G, $V = \{1,2, \dots, n\}$ and E is edge, $E \subseteq V \times V$. The neighbors of node i is denoted by N_i

$$N_i = \{j_1, j_2, \dots, j_n\}$$

For example, in figure 3.1

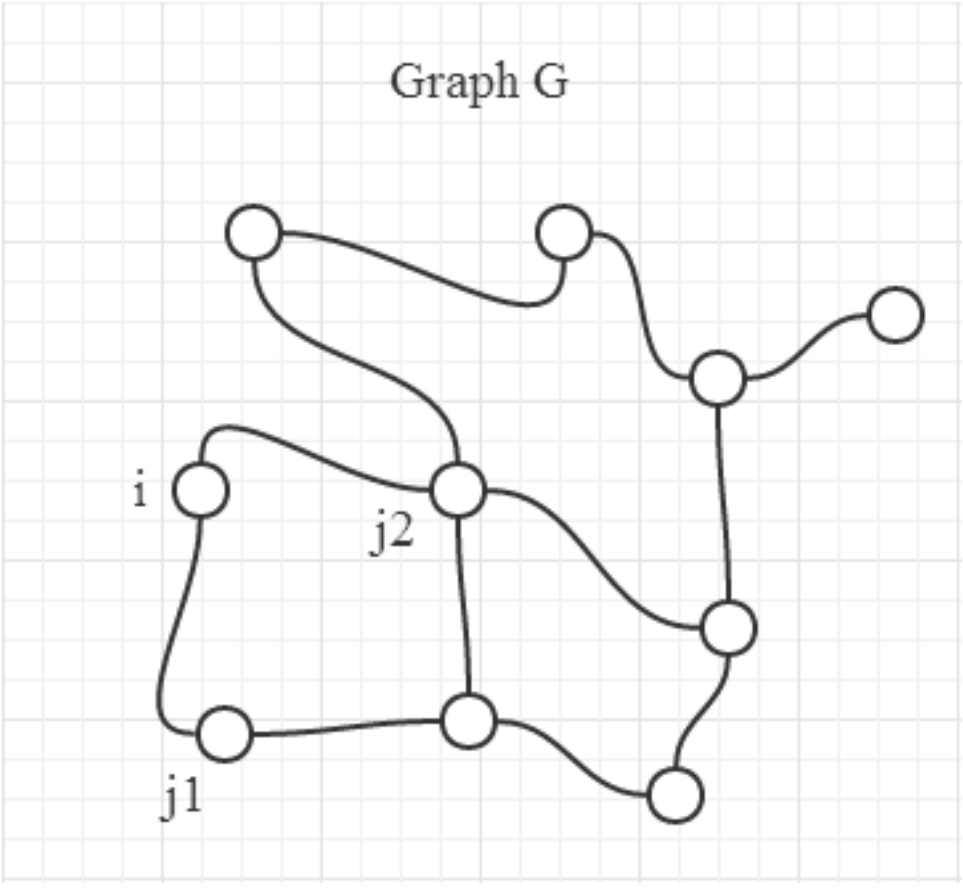

Figure 3.1: Undirected graph G

The consensus problem is to ensure that all the nodes in the graph reach a common

value, in our case, the average of the initial values. When the network has a centralized node, it is easy to achieve this. Every node sends the initial value to the central node. Central node calculates the average and sends it to all the other nodes. But if there is no central node, we need a distributed average consensus algorithm.

For solving the above problem, there are typically two algorithms: probabilistic counting and pairwise averaging [1]. In this thesis we only discuss about pairwise averaging.

Basically each node sends its number to its neighbors. The neighbors carry the informations and exchange with other nodes, the states of these nodes will change. It is a distributed consensus algorithm which guarantees convergence to an agreement value via interactions between nodes [4]. Asymptotically all nodes will carry the agreement value after enough communications. This value is the average of these nodes in initial state. This is similar to mixing a cup of hot water to other cups of cool water; finally the water of all cups will be warm.

3.1 Applications

Consensus problem of multiple agents system has many applications in various areas of science and engineering.

One example is Synchronization of Coupled Oscillators, which is a topic in the field of biology, physics and even mathematics. For generalized Kuramoto model, coupled oscillators can be shown as $\dot{\theta}_i = k\sum_{j\in N_i}\sin(\theta_j - \theta_i) + \omega_i$, where θ is phase and ω is frequency. For all initial states, synchronization can be reached in the network, if k is large enough. The convergence analysis can help answer questions like how to achieve

synchronization in oscillator networks [3].

Another example is Flocking Theory on distributed sensing of group of mobile agents with communication devices in networks. The flock is a network of the dynamic system and it is a state dependent graph in topology. The consensus algorithm is based on velocity matching between agents and their neighbors, and it relies on the state of every agent [3].

One important problem on consensus is the converge rate; for example, how quickly to achieve the consensus agreement by using semi definite convex programming. Since the algebraic connectivity determines the speed of convergence, to get faster consensus algorithm, multiple orders are used to increase algebraic connectivity [3].

Rendezvous in space is another application of consensus problems. By reaching a consensus of group agents, agreement space is determined with a large number of agents in a network. This rendezvous is called “unconstrained consensus problem” [3].

Consensus can also be applied to Distributed Control of multi-agent systems. In military and commercial field, there are two kind of distributed formation controls. Representation of formations is one of them. This is a theoretical framework for designing and analyzing distributed controller. The theory of consensus problem and analyses are described in [3].

This thesis discusses the distributed average problem. In dynamic system, distributed average problem requires linear least square estimator to carry out the consensus average. Consensus filter is developed for calculating average of this system [3].

3.2 Algorithm

Assume the graph is undirected $a_{ij} = a_{ji}$ and no external intervention, so the sums of all nodes at $t = 0$ and $t = \infty$ are the same [3].

$$S = \sum_i x_i(0) = \sum_i x_i(\infty) \tag{1}$$

$$A = \frac{1}{n}\sum_i x_i(0) \tag{2}$$

And for undirected graph, it follows,

$$\dot{x}_i(t) = \sum_{j \in N_i} a_{ij}\left(x_j(t) - x_i(t)\right) \tag{3}$$

$$\sum_i \dot{x}_i(0) = 0 \tag{4}$$

For this dynamic system, $\dot{x}$ can be expressed as

$$\dot{x} = -Lx \tag{5}$$

L is the graph Laplacian of G. It is defined as

$$L = D - A \tag{6}$$

$$D = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_i \end{bmatrix} \tag{7}$$

D is degree matrix of G, and

$$d_i = \sum_{j \neq i} a_{ij} \tag{8}$$

For undirected graph, one property of Laplacian is “sum of squares”

$$x^T Lx = \frac{1}{2} \sum_{(i,j)\epsilon E} a_{ij} \left(x_j - x_i\right)^2 \tag{9}$$

Define a quadratic disagreement function

$$\varphi(x) = \frac{1}{2} x^T Lx \tag{10}$$

So $\dot{x}_i(t)$ is shown as

$$\dot{x} = -\nabla\varphi(x) \tag{11}$$

It has been proved in [3].

It means if a consensus is asymptotically achieving agreement, collective decision is equal to the average of all nodes at $\mathrm{t} = 0$ (initial state). This consensus algorithm is called average consensus algorithm in distributed computing [5]. If there is no external intervention, average A follows result

$$A = \frac{1}{n} \sum_{i\in n} x_i(0) = \frac{1}{n} \sum_{i\in n} x_i(\infty) \tag{12}$$

In this paper, the implementation network consist of 3 devices, so the nodes are x_1, x_2, and x_3. Suppose the states of 3 nodes change every certain seconds and the two connected nodes exchange their numbers, then both nodes calculate average over their last states.

At initial state

$$\begin{aligned} x_1(0) &= a \\ x_2(0) &= b \\ x_3(0) &= c \end{aligned} \tag{13}$$

Sum and average are

$$S = \sum_i x_i\,(0) = a + b + c \tag{14}$$

$$A = \frac{1}{n}\sum_i x_i\,(0) = \frac{a+b+c}{3} \tag{15}$$

Next state expectations of 3 devices are

$$\begin{aligned} x_1(1) &= \frac{1}{2}\frac{a+b}{2} + \frac{1}{2}\frac{a+c}{2} \\ &= \frac{2a+b+c}{4} \\ x_2(1) &= \frac{1}{2}\frac{a+b}{2} + \frac{1}{2}\frac{b+c}{2} \\ &= \frac{2b+a+c}{4} \\ x_3(1) &= \frac{1}{2}\frac{a+c}{2} + \frac{1}{2}\frac{b+c}{2} \\ &= \frac{2c+a+b}{4} \end{aligned} \tag{16}$$

For x_1 at iterative t

$$x_1(t) = \frac{\left(4^t - \frac{2(4^t-1)}{4-1}\right)a + \left(4^t - \frac{2(4^t-1)}{4-1} - 1\right)(b+c)}{4^t} \tag{17}$$

If $t \to \infty$

$$\begin{aligned} x_1(t) &= \frac{\frac{4^t}{3}(a+b+c)}{4^t} \\ &= \frac{x_1(0) + x_2(0) + x_3(0))}{3} \\ &= A \end{aligned} \tag{18}$$

Similar for x_2 and x_3 there are the same result. At iterative t, the value of one node

is equal to average of all nodes, if t is large enough.

Next, I will present the implementation of the above algorithm.

CHAPTER 4

PROGRAM STRUCTURE

To develop the simulate program, only three devices are initialized. Suppose there are 3 PDA which are denoted by A, B, C and they all have Bluetooth services. Each unit has an initial value at the beginning. Each of them can connect to the others but the connection is random. If successful connecting, every unit will pick one from the other two units randomly and send it the value (number). When device A receives the number from B, and it also picks B to send value. Node A calculates between the value of B and its own and updates its own value to the average. B does the same process. Now both A and B have the same value which are the averages of their last value.

System runs the loop under control of time, until the 3 numbers achieve a theoretical value in specified precision. Then program records procedure, loop number and data in a TXT file.

4.1 Synchronous and Asynchronous

In a simple Bluetooth program, slave devices need to receive message almost all time and only few time for sending message. There are two ways to implement Bluetooth communication: single thread and multiple threads. Single thread system blocks other procedures while receiving or sending message. Note for a single thread system, when a device sends a value to another, the system will hold until a response [19], [20]. However, in this thesis, unblock for sending and receiving procedure is necessary. Systems can continue going to next part without receiving a number or response. Simulation of exchanging data

needs to separate sending and receiving into two independent procedures. They are asynchronous, so that multiple threats are needed.

4.2 Initial System

Figure 4.1 shows the interface of the program. The buttons include “Server”, “Connect”, “Send”, and “Auto”. The initial value can be also input through the user interface.

The dialog and button are implemented through function of “MainDlgProc (HWND, UINT, WPARAM, LPARAM)”. Then the function of “SetTimer (hDlg, 1, 1000, NULL)” creates timer and show time on the top of dialog.

The values of three PDAs and theoretical value must be real numbers. After running the program, both transmitted and received message are shown in the main window. The iterative number will be displayed on the cell of “Loop number”. Note there is a “cell” labelled as “Theoretical value”, which is the theoretical average value to be approached.

When the values of nodes reach the theoretical value, it will stop the consensus process. If no value indicated, the node will run the program forever unless manually stop.

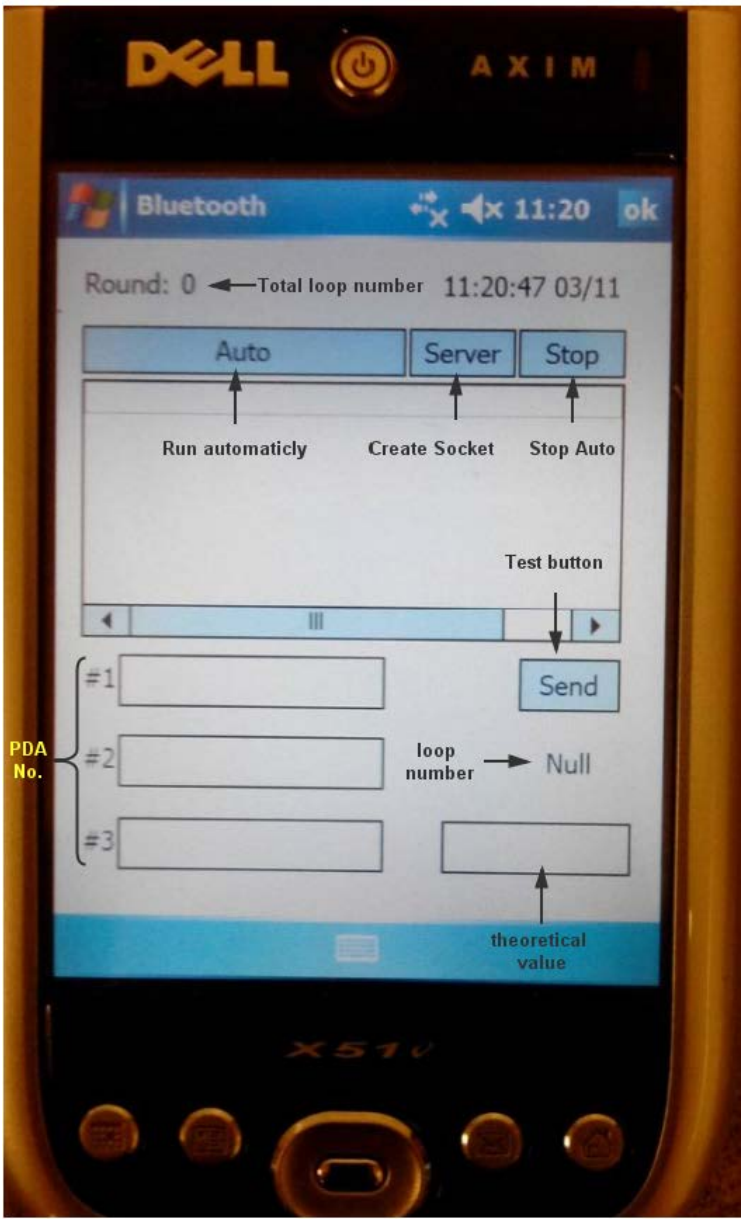

Figure 4.1: Program interface

4.3 Create Socket

The socket communication is the main component for the Bluetooth connection. The function of “socket (AF_BTH, SOCK_STREAM, RFCOMM)” creates the interface of the program [19].

The function includes address family, type, and protocol. Address family is always AF_BTH, type is set as STREAM, and protocol is set as RFCOMM. These three parameters

decide that the socket works for Bluetooth and cannot be changed [18].

Through windows API, the set of creating Bluetooth socket can be done by clicking the “Server” button on the screen of PDA.

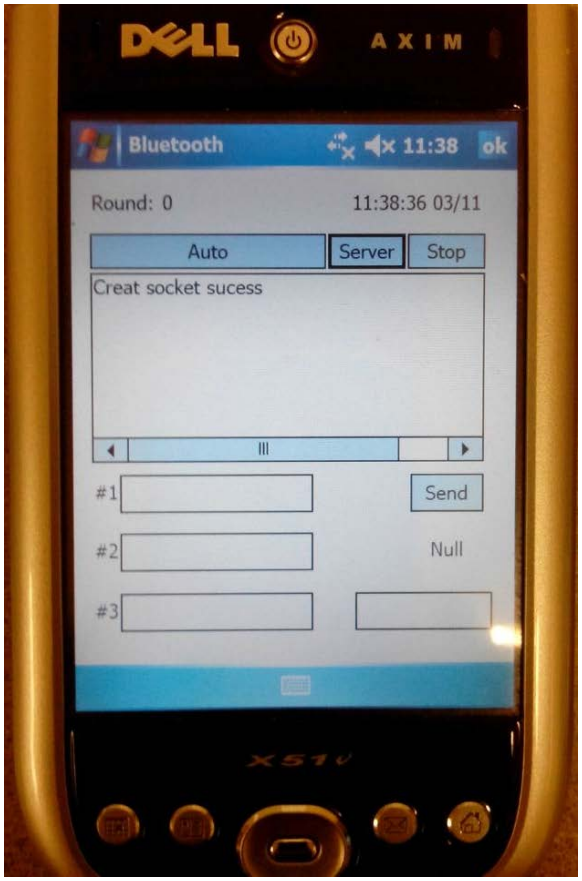

Figure 4.2: Create socket success

4.4 Time Control

The consensus is executed in synchronized way. Every 15 second there is one iteration. All the three PDAs are synchronized with the same time slots. If device does not send or receive any valid message, system will print “Time out” on the screen then return to the beginning of the loop. The system will continue running unless “Stop” button is clicked.

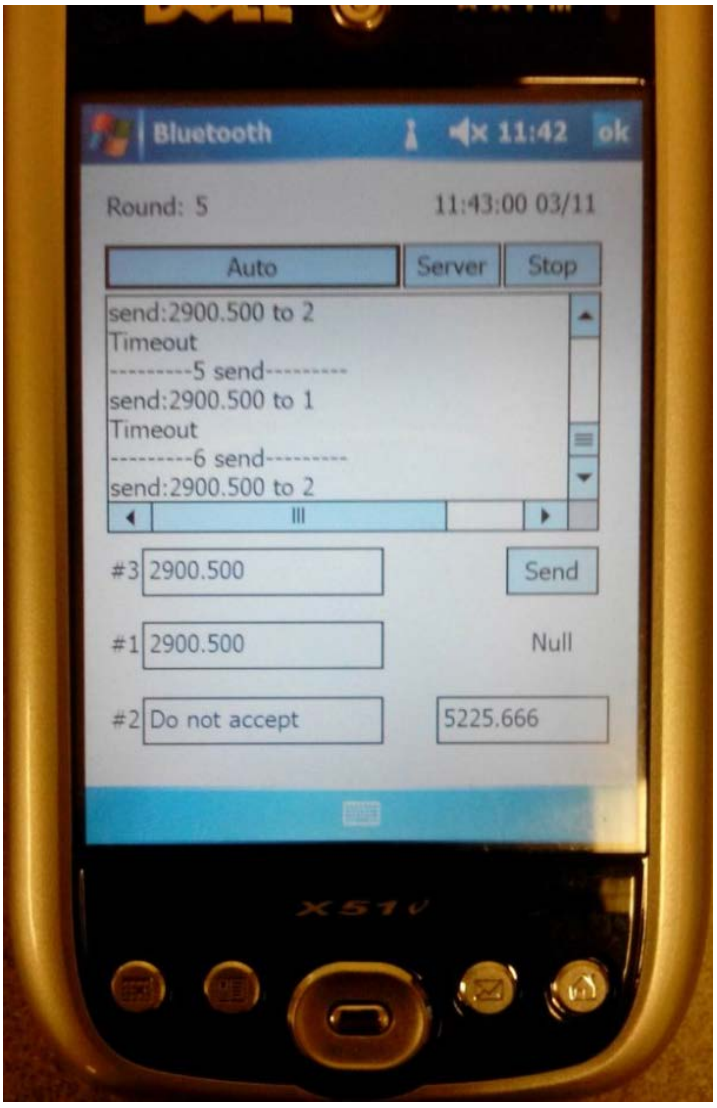

Figure 4.3: Time out

4.5 Generate Random Number

In C, random number is generated by “srand ()” and “rand ()”. The function “srand ()” offers seeds for “rand ()” to generate random number. However if seeds are the same, the series numbers are totally same. Not only are the random numbers of 3 PDAs different, every time the program starts, the number of each PDA should be different from itself. Seeds must be replaced every time before generating random number. If using arithmetic progression to change seed, order of numbers obey some regular patterns.

Fibonacci sequence is an integer sequence which the third number equals to sum of first number and second number. We can set first and second number any number, and third number will not be same in each loop and each PDA.

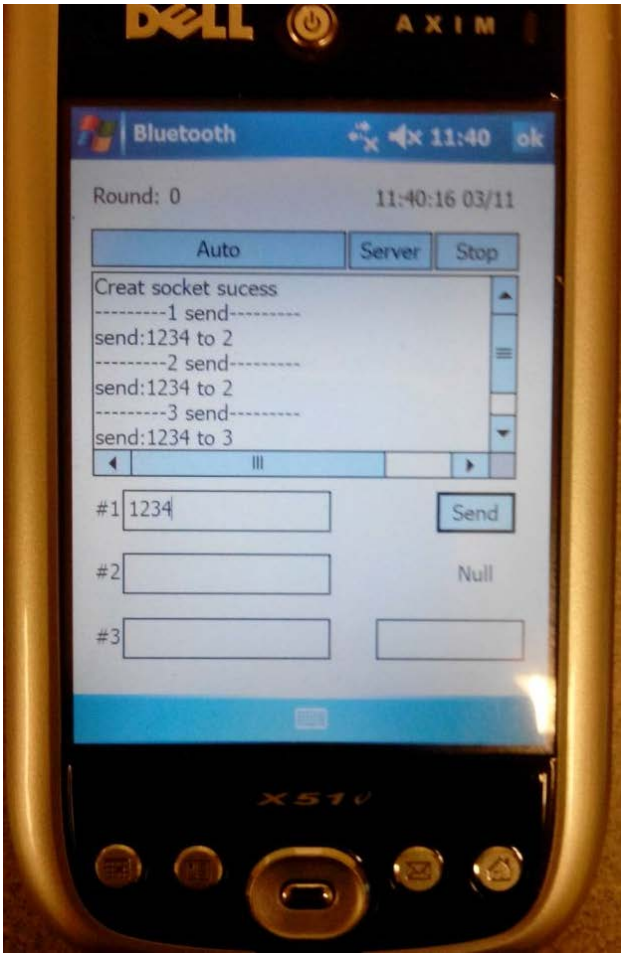

Figure 4.4: Send message randomly

As shown in figure 4.4, after generating random number, PDA1 sends message to PDA2 or PDA3 randomly

4.6 Connect and Error

The socket connection is implemented by the function of “connect (SOCKET, sockaddr, namelen);”. If connection succeeds, the socket name number will be returned. Otherwise, an error message with error code will be returned and shown on screen. In the

function, “SOCKET” is used to describe and identify the unconnected socket. It must be valid and unused. “sockaddr” is a pointer to Bluetooth address structure. “namelen” is integer number to describe length of Bluetooth address by bytes.

If connection fails, the function will return error 10060 or 10061 which means connection time out or connection refused [18], [19].

In addition, the three devices will begin to connect to others at the same time, because Bluetooth system in this program receives one connect application at same time, connections may conflict. Therefore, when the error code “10061” received, it only means there is no connection at this iteration. The whole consensus program is still running.

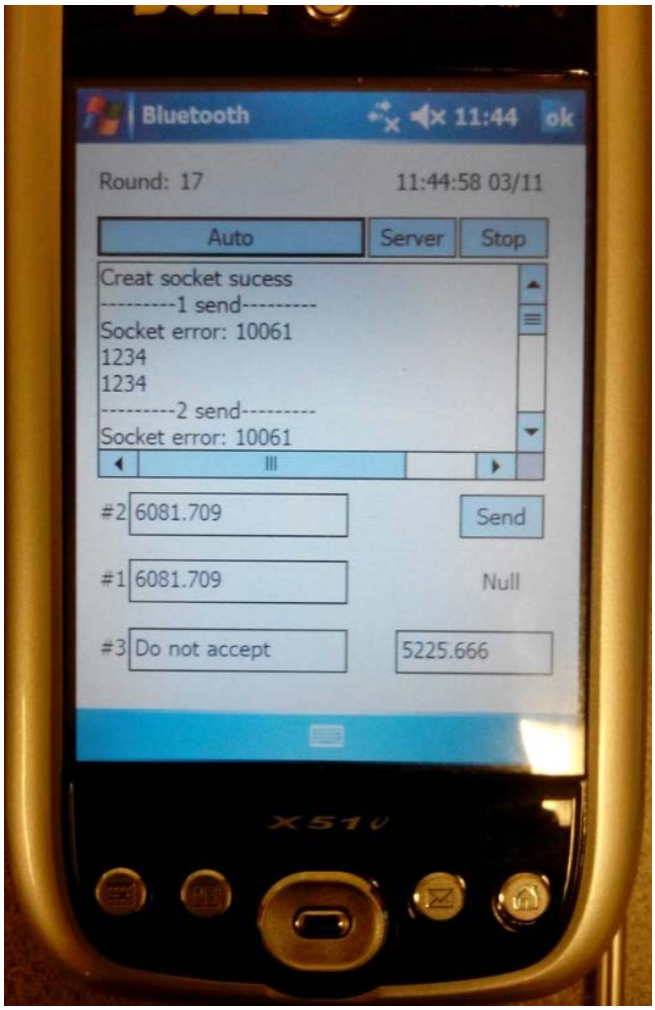

Figure 4.5: Socket error

4.7 How to Exchange Data

After successful connection, devices can send or receive message from other devices which are running two functions:

“send (s, buf, len, flags);”

“recv (s, buf, len, flags);”

In both functions, “s” is a sign to identify connected socket. “buf” is a pointer to keep transmitted data. “len” is the length of the data by bytes. “flags” specifies the way to send or receive. Normally it is 0 in this program.

After generating random number, one device, called A, intentionally pick a device from others. Then A sends its value to targeting device B. If B also picks A, B also sends B's value to A.

After sending and receiving data, system will wait response from the targeting device. If receiving data is successful, system calculates the average. If not, system prints "Time out" and waits for next loop.

Figure 4.6 shows the sending and receiving processes. In this figure, the PDA 1 pairs with PDA 2 for information exchanging.

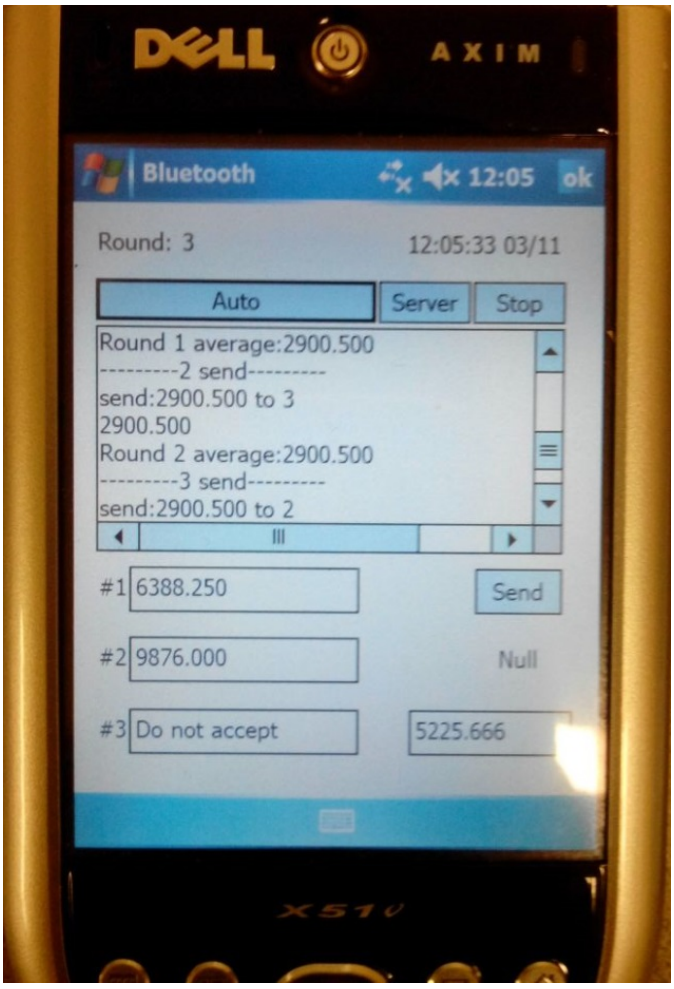

Figure 4.6: Exchange success and calculate average

4.8 Data Calculation and Recording

Date is kept in float type with 3 digits after the decimal point. In this program, negative number is not supported.

“WriteFile ()” function records all data at the same time. Log file is in TXT format. It includes iteration number, received message, transmitted message, average number, and whether pairs with other device or not.

Clicking “Ok” button can stop program and return to desktop. All temporary data, except recorded data, will be cleared.

When average achieves the expect value, system will save the iteration number. In figure 4.7, PDA 1 and PDA 2 achieve the value at 42th loop. PDA 3 did not pair with 1 or 2 at 42-49th loops, so it recorded 50th loop. The theoretical value on the bottom of window is only for checking the result to indicate that iteration that needed to achieve the average.

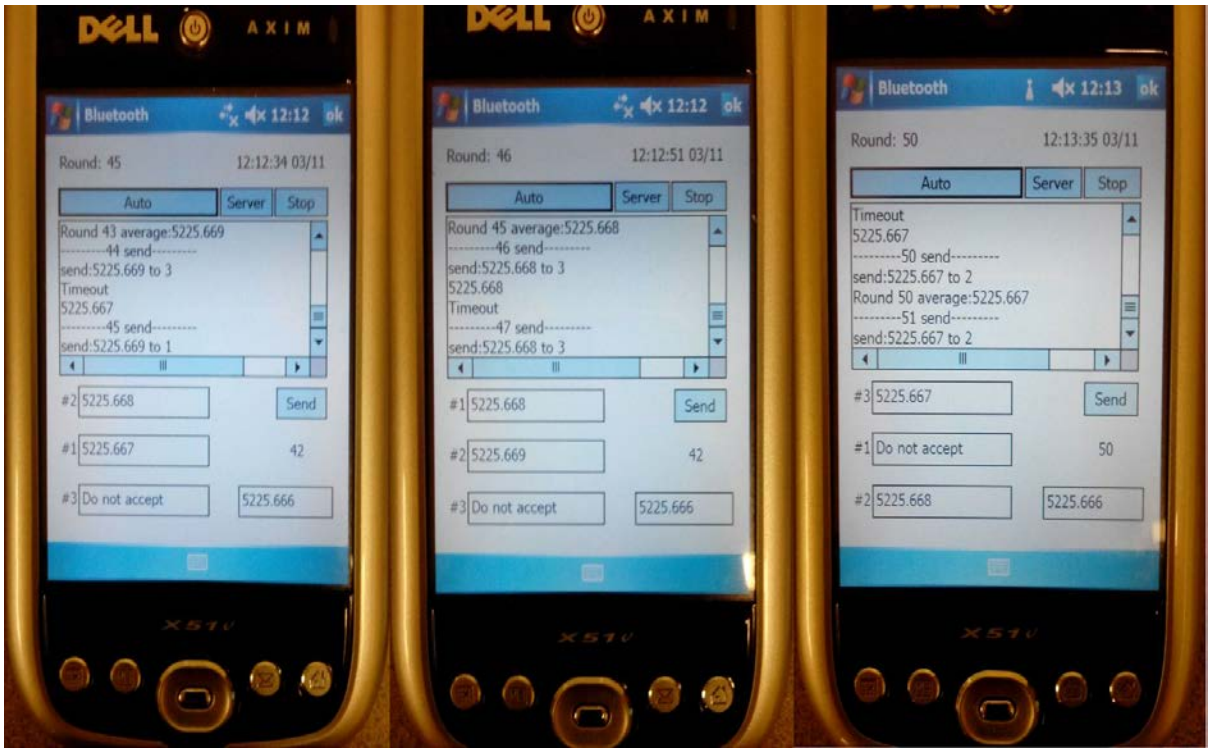

Figure 4.7: Screen shot of one experiment

Figure 4.8 shows the log file. First column shows the iteration number and the value of the PDA in current state. The second column shows the target PDA. Third and fourth are the received message and the targeting PDA number. The last column shows the current average and the pair informations.

The data is respectively as the form in figure 4.8.

File Edit Format View Help

```
==========Begin time:16:40:46 02/05==========
1:9674.000      Send to #3:9674.000     Receive from #1:Null    #3:Null         Result:9674.000  Pair with None
2:9674.000      Send to #3:9674.000     Receive from #1:Null    #3:Null         Result:9674.000  Pair with None
3:9674.000      Send to #1:9674.000     Receive from #1:Null    #3:Null         Result:9674.000  Pair with None
4:9674.000      Send to #1:9674.000     Receive from #1:Null    #3:Null         Result:9674.000  Pair with None
5:9674.000      Send to #1:9674.000     Receive from #1:Null    #3:Null         Result:9674.000  Pair with None
6:9674.000      Send to #3:9674.000     Receive from #1:Null    #3:2409.500     Result:6041.750  Pair with 3
7:6041.750      Send to #3:6041.750     Receive from #1:Null    #3:6041.750     Result:6041.750  Pair with 3
8:6041.750      Send to #1:6041.750     Receive from #1:2409.500#3:Null         Result:4225.625  Pair with 1
9:4225.625      Send to #1:4225.625     Receive from #1:Null    #3:Null         Result:4225.625  Pair with None
```

Figure 4.8: Log file

4.9 Overflow and Precision

Because the data type of value is in “int”, the range of the value should be in (0, 2147483647). Using number out of range may cause the system unstable and leads to wrong results. The cell of float number precision is set 0.001. It means if absolute difference between the average number and theoretical value is less than 0.001, the system considers the two values are the same. In this case, the consensus average is achieved.

4.10 Program Structure

Figure 4.9 shows the structure of the entire program. System initials Bluetooth at the beginning, then it creates Winsock API. API offers access ports to connection. After connection, system generates stochastic number to decide which PDA is picked. PDA sends messages to picked device and waits for response if no error occurs. Otherwise, system will return to “connect” step. If system receives a valid data, it will calculate average and generate the output. At last average and preinstall value are compared. If they are equal (achieve precision), system will go to “end”.

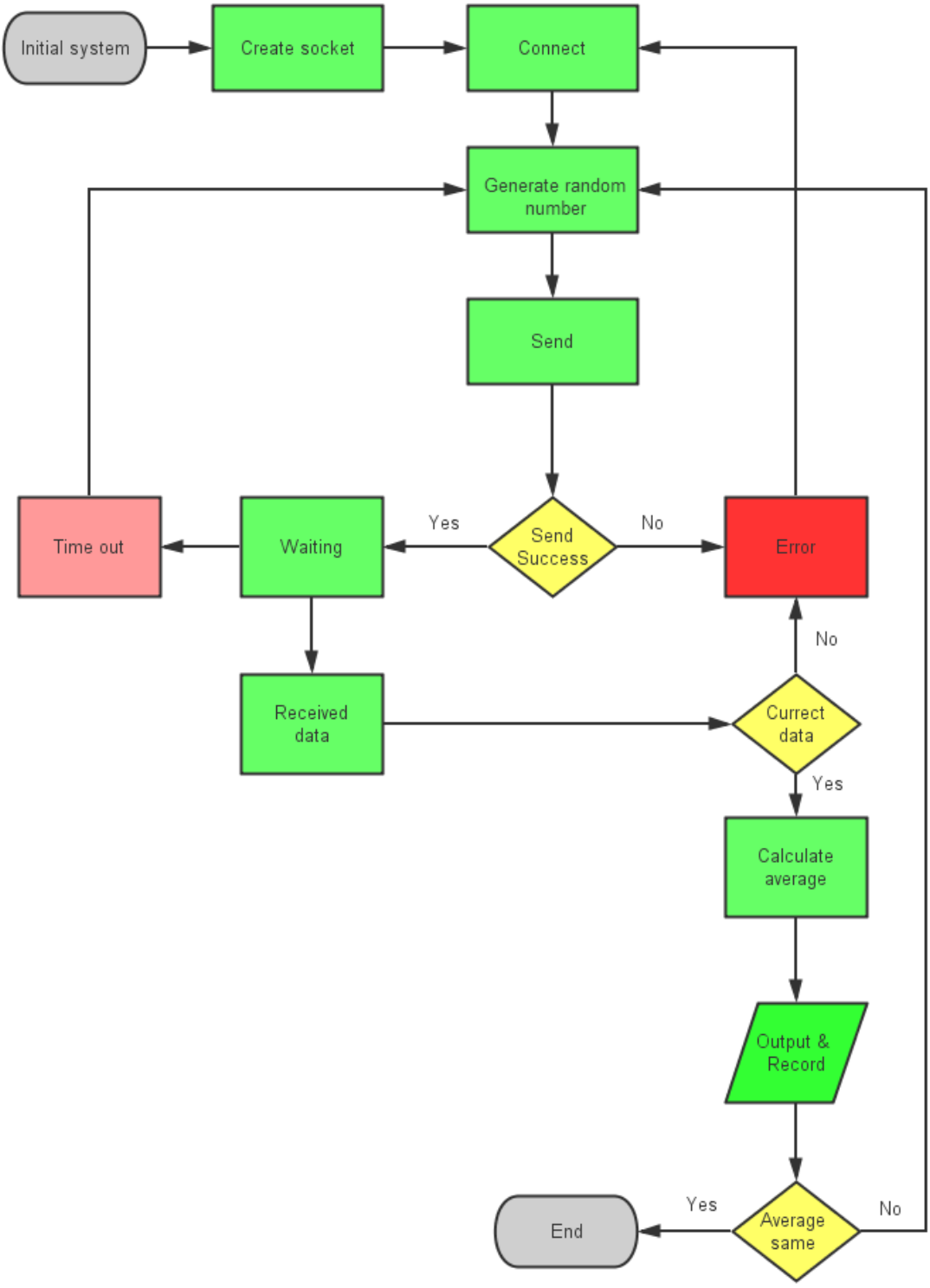

Figure 4.9: Program structure

CHAPTER 5

CONCLUSION

In this thesis, the distributed consensus algorithm is implemented. A three nodes small network is presented to show information exchanging through Bluetooth channels. Experimental results show that consensus is achieved and all the three nodes can approach to the theoretical average, which validates the theoretical analysis. The thesis discusses some challenges to implement the simple average consensus, such as synchronization, setting up communication through Bluetooth pairing and random number generation. The thesis does not include some other considerations such as channel collisions and radio spectrum change. Also the program skips the inquiry and page state, because the physical addresses have already been built in the program.

APPENDIX

CODE STRUCTURE AND RESULT

Code structure

Some parts of main.cpp

```
Main.cpp

INT_PTR CALLBACK MainDlgProc(HWND hDlg, …)
{
    switch (message)
    {
        case WM_INITDIALOG:
        {
            /*Create dialog*/
            SHINITDLGINFO window;
            window.dwMask = SHIDIM_FLAGS;
            window.dwFlags = SHIDIF_DONEBUTTON |
                             SHIDIF_SIZEDLGFULLSCREEN |
                             SHIDIF_EMPTYMENU;
            window.hDlg = hDlg;/*handle of window*/
            …
            …

            SetTimer(hDlg,1,1000,NULL); /*timer*/

            /*create a text to show loop number*/
            SetWindowText (GetDlgItem (hDlg, …);

            /*Create txt file to record data*/
            fileHandle = CreateFile(L"\\MyDocuments\\
                         Business\\record.txt",…
            SetFilePointer (fileHandle, 0, NULL, FILE_END);
            WriteFile (fileHandle, …);

            /*fill PDA number and button name in the dialog*/
            StringCchPrintf(initi, …);
            SetWindowText(GetDlgItem(hDlg, …);
             …
             …
        }
        /*functions of buttons*/
        case WM_COMMAND:
           switch(LOWORD(wParam))
```

```
        {
            /*Stop button for ending auto thread*/
            case IDC_STOP:
                if (h) /*h is handle of average function*/
                {
                    TerminateThread (h, 0);
                 }
                 break;

            /*server button to create socket for Bluetooth by
              call OpenServer function in Bth class*/
            case IDC_SERVER:
                i=objBthUtils.OpenServer(
                        rgbSdpRecord, …);
                if(i!=0)
                {
                    /*create fail return message on screen*/
                    StringCchPrintf (szMessage, …);
                    SendMessage(GetDlgItem(hDlg, …);
                }
                else  /*create success*/
                {
                    StringCchPrintf(szMessage, …);
                    SendMessage(GetDlgItem(hDlg, …);
                }
                break;

            /*send button for testing*/
            case IDC_SEND:
                SendMessage(GetDlgItem(hDlg, …);
                SendData(hDlg);  /*function in bth class*/
                …
                break;
            /*auto button  program will run automatically*/
            case IDC_AUTO:
                /*create new thread for autorun function*/
                h=CreateThread(NULL, 0, autorun, (LPVOID)hDlg,
                               0, NULL);
                break;

            /*click ok button to over program*/
            case IDOK:
```

```
                /*close dialog, end thread and timer*/
                EndDialog(hDlg, LOWORD(wParam));
                if (h)
                {
                   TerminateThread(h, 0);
                 }
                 KillTimer(hDlg,1);
                 break;
                 }
      break;
      /*show time on top of dialog*/
      case WM_TIMER:
         GetLocalTime(&t);
         StringCchPrintf(timebuf, …);
         SetDlgItemText(g_hDlg, …);
         break;
      /*when dialog is ended windows will close*/
      case WM_CLOSE:
         EndDialog(hDlg, message);
         return TRUE;
         break;
   }
return (INT_PTR)FALSE;
}

/*Autorun function*/
DWORD WINAPI autorun(LPVOID voidDlg)
{
   objBthUtils.randomnum = 0;              /*random number*/
   HWND hDlg = (HWND) voidDlg;   /*handle of receive thread*/
   …
   …
   while(1)
   {
      GetLocalTime(&t2);                  /*get system time*/
      int timec=t2.wSecond%10;
      if (timec==0)                      /*if have not sent*/
      {
         if (sendtimes==0)
         {

            /* send message to target device and show on screen*/
```

```
            SendMessage(GetDlgItem(hDlg, IDC_DEVICES), …);
            SendData(hDlg);
            …
            sendtimes=1;                /*sign it has sent*/
         }
         Sleep(1500);             /*system sleep for 15 sec*/
         while(1)
         {
            if (objBthUtils.randomnum==1)   /*pick target 1*/
            {
               AverageValue(hDlg);           /*get average*/
               recflag=0;                   /*reset recflag
               sendtimes=0;                     &sent flag*/
               break;
            }
            if (objBthUtils.randomnum==2)   /*pick target 2*/
            {
               AverageValue(hDlg);
               …
               …
            }

            /*get system time again*/
            GetLocalTime(&t1);
            /*if have not receive any response*/
            if(t1.wSecond-t2.wSecond>=8)
            {
               PostMessage(…, "Timeout");
               NoPairRecord(hDlg);        /*record no pair*/
               recflag=0;      /*reset recflag & sent flag*/
               sendtimes=0;
               break;
            }
         }
      }
   }
}

/*AverageValue function, data is calculated in this function*/
void AverageValue(HWND hDlg)
{
   int numloop, …;               /*some definition of variable*/
```

```
    float f, …;
    WCHAR wchc[MAX_MESSAGE_SIZE], …;
    GetDlgItemText(hDlg, …);     /*get data from receive buff*/
    …
    …

    /*receive data is wchar, convert it to float*/
    wcharTochar(message, …);
    fla = atof();
    …
    …

    /*get average with target PDA*/
    if (objBthUtils.randomnum==1)        /*if target is PDA 1*/
    {
       flaverage=(fla+flc)/2;
       StringCchPrintf(wREC1, …); /*convert data back to wchar*/
       SetDlgItemText(g_hDlg, …);       /*post data on screen*/
       pdarec=1;                                  /*set flags*/
    }
    if (objBthUtils.randomnum==2)        /*if target is PDA 2*/
    {
       flaverage=(flb+flc)/2;
       StringCchPrintf(wREC2, …);
       SetDlgItemText(g_hDlg, …);
       pdarec=2;
    }
    …
    …

    /*account number of loop*/
    numloop=numloop+1;

    /*record function*/
    record (…);
    …
    …

    /*if reach precision, system will record all data*/
    if ((flaverage-fle)<=0.001&&(flaverage-fle)>=-0.001)
    {
       SetFilePointer (fileHandle, 0, NULL, FILE_END);
```

```
        sprintf(buff, …);
        WriteFile(fileHandle, …);
    }
}

/* message is sent by using this function*/
void SendData(HWND hDlg)
{
    …
    if(iRetVal)                          /*if socket is not exist*/
    {
        StringCchPrintf(szMessage, …);   /*return error message*/
    }
    Else                                        /*socket exist*/
    {
        if (objBthUtils.randomnum==1)      /*send to target 1*/
        {
            SendMessageto (…);
        }
        if (objBthUtils.randomnum==2)      /*send to target 2*/
        {
            SendMessageto (…);
        }
        …
    }
    …
}
```

Some part of component.cpp

```
/*Create socket to receive connection application*/
int ServerConnect(BYTE *Record,…)
{
   if(socketSer==INVALID_SOCKET) /*check whether socket is used*/
   {
       socketSer = socket (AF_BT, …);   /*create socket server*/
       if (m_socketSer  == INVALID_SOCKET)
       {
           return WSAGetLastError ();      /*return error code*/
       }
       /*socket address information*/
```

```
        SOCKADDR_BTH ser;
        memset (&ser, 0, sizeof(sa));
        ser.addressFamily = AF_BT;
        ser.port = 0;

        /*bind and listen to this address*/
        if (bind (socketSer, …)
        {
            return WSAGetLastError ();
        }
        …
        …
        if (listen (socketSer, SOMAXCONN))
        {
            return WSAGetLastError ();
        }
    }
    return 0;
}

/* connect to other device*/
int ClientConnect (WCHAR *GUID,…)
{
    if (socketCli==INVALID_SOCKET)/*check whether socket is used*/
    {
        …
        GetGUID(GUID, &ServerGuid);                /*get GUID*/
        socketCli = socket (AF_BT, …);     /*create socket client*/
        if (socketCli == INVALID_SOCKET)
        {
            return WSAGetLastError();
        }

        /*address information of target device*/
        s.addressFamily = AF_BT;
        s.serviceClassId=ServerGuid;
        s.btAddr=0X0012d12b6dc3;

        /*connect fail or success*/
        if (connect (socketCli, (SOCKADDR *)&s, sizeof(s)) ==
            SOCKET_ERROR)
        {
```

```
            socketCli=INVALID_SOCKET;
            return WSAGetLastError();
        }
    }
    return 0;
}

/*generate random number*/
int randnumber(int m1, int m2)
{
    SYSTEMTIME a;              /*seed include two part: system
    GetLocalTime (&a);              time and series number m3*/
    m3 = m1+m2
    srand(m3+a.wSecond%10);
    rnum=rand()%2+1;
    m2=m1;                         /*next time m3 is changed*/
    m3=m2;
    return rnum;
}

/* send message function*/
int SendMessageto(WCHAR *GUID, …)
{
    /*get random number from randnumber()*/
    randomnum=randnumber();
    if(m_socketClient1==INVALID_SOCKET)
    {

        /*connect to other device*/
        Connectflag=ClientConnect(GUID, …);
        …
        if(Connectflag!=0)                  /*connect problem*/
        {
            return Connectflag;
        }
        else
        {
            /*pick one address to send*/
            if (randomnum==1)
            {
                …
                send (socketCli,...);
```

```
            }
            if (randomnum==2)
            {
                …
                send (socketCli, ...);
            }
            …
        }
    return 0;
}
```

Other file such as head file “stdafx.h” is generated by system.

Result
The result in this case is:
PDA 1’s value is 1234
PDA 2’s value is 9876
PDA 3’s value is 4567

For PDA 1

```
======================Begintime:12:01:58 03/11=========================
1:1234.000   Send #3:1234.000 Receive #2:Null #3:4567.000 Result:2900.500       Pair  with
3
2:2900.500   Send #3:2900.500 Receive #2:Null #3:2900.500 Result:2900.500       Pair  with
3
3:2900.500   Send #2:2900.500 Receive #2:9876.000 #3:Null Result:6388.250       Pair  with
2
4:6388.250   Send #2:6388.250 Receive #2:6388.250 #3:Null Result:6388.250       Pair  with
2
5:6388.250   Send #2:6388.250 Receive #2:6388.250 #3:Null Result:6388.250       Pair  with
2
6:6388.250   Send #2:6388.250 Receive #2:Null #3:Null Result:6388.250   Pair with None
7:6388.250   Send #2:6388.250 Receive #2:Null #3:Null Result:6388.250   Pair with None
8:6388.250   Send #3:6388.250 Receive #2:Null #3:Null Result:6388.250   Pair with None
9:6388.250   Send #3:6388.250 Receive #2:Null #3:4644.375 Result:5516.313       Pair  with
```

3
10:5516.313 Send #2:5516.313 Receive #2:4644.375 #3:Null Result:5080.344 Pair with 2
11:5080.344 Send #3:5080.344 Receive #2:Null #3:Null Result:5080.344 Pair with None
12:5080.344 Send #2:5080.344 Receive #2:5080.344 #3:Null Result:5080.344 Pair with 2
13:5080.344 Send #3:5080.344 Receive #2:Null #3:5516.313 Result:5298.329 Pair with 3
14:5298.329 Send #2:5298.329 Receive #2:5080.344 #3:Null Result:5189.337 Pair with 2
15:5189.337 Send #3:5189.337 Receive #2:Null #3:Null Result:5189.337 Pair with None
16:5189.337 Send #3:5189.337 Receive #2:Null #3:5298.329 Result:5243.833 Pair with 3
17:5243.833 Send #3:5243.833 Receive #2:Null #3:5243.833 Result:5243.833 Pair with 3
18:5243.833 Send #2:5243.833 Receive #2:Null #3:Null Result:5243.833 Pair with None
19:5243.833 Send #3:5243.833 Receive #2:Null #3:5243.833 Result:5243.833 Pair with 3
20:5243.833 Send #2:5243.833 Receive #2:5189.337 #3:Null Result:5216.585 Pair with 2
21:5216.585 Send #3:5216.585 Receive #2:Null #3:5243.833 Result:5230.209 Pair with 3
22:5230.209 Send #3:5230.209 Receive #2:Null #3:Null Result:5230.209 Pair with None
23:5230.209 Send #3:5230.209 Receive #2:Null #3:Null Result:5230.209 Pair with None
24:5230.209 Send #3:5230.209 Receive #2:Null #3:5223.397 Result:5226.803 Pair with 3
25:5226.803 Send #2:5226.803 Receive #2:5223.397 #3:Null Result:5225.100 Pair with 2
26:5225.100 Send #3:5225.100 Receive #2:Null #3:5226.803 Result:5225.952 Pair with 3
27:5225.952 Send #3:5225.952 Receive #2:Null #3:5225.952 Result:5225.952 Pair with 3
28:5225.952 Send #2:5225.952 Receive #2:Null #3:Null Result:5225.952 Pair with None
29:5225.952 Send #3:5225.952 Receive #2:Null #3:Null Result:5225.952 Pair with None
30:5225.952 Send #3:5225.952 Receive #2:Null #3:Null Result:5225.952 Pair with None
31:5225.952 Send #3:5225.952 Receive #2:Null #3:5225.526 Result:5225.739 Pair with 3

32:5225.739 Send #3:5225.739 Receive #2:Null #3:5225.739 Result:5225.739 Pair with 3
33:5225.739 Send #2:5225.739 Receive #2:5225.526 #3:Null Result:5225.632 Pair with 2
34:5225.632 Send #2:5225.632 Receive #2:5225.632 #3:Null Result:5225.632 Pair with 2
35:5225.632 Send #2:5225.632 Receive #2:5225.632 #3:Null Result:5225.632 Pair with 2
36:5225.632 Send #2:5225.632 Receive #2:Null #3:Null Result:5225.632 Pair with None
37:5225.632 Send #2:5225.632 Receive #2:Null #3:Null Result:5225.632 Pair with None
38:5225.632 Send #3:5225.632 Receive #2:Null #3:5225.739 Result:5225.686 Pair with 3
39:5225.686 Send #2:5225.686 Receive #2:5225.632 #3:Null Result:5225.659 Pair with 2
40:5225.659 Send #3:5225.659 Receive #2:Null #3:Null Result:5225.659 Pair with None
41:5225.659 Send #3:5225.659 Receive #2:Null #3:Null Result:5225.659 Pair with None
42:5225.659 Send #2:5225.659 Receive #2:5225.673 #3:Null Result:5225.666 Pair with 2
Achieve at 42 ----------
43:5225.666 Send #3:5225.666 Receive #2:Null #3:Null Result:5225.666 Pair with None
44:5225.666 Send #3:5225.666 Receive #2:Null #3:5225.669 Result:5225.667 Pair with 3
45:5225.667 Send #2:5225.667 Receive #2:5225.669 #3:Null Result:5225.668 Pair with 2
46:5225.668 Send #3:5225.668 Receive #2:Null #3:Null Result:5225.668 Pair with None
47:5225.668 Send #3:5225.668 Receive #2:Null #3:5225.667 Result:5225.667 Pair with 3
48:5225.667 Send #2:5225.667 Receive #2:Null #3:Null Result:5225.667 Pair with None
49:5225.667 Send #2:5225.667 Receive #2:Null #3:Null Result:5225.667 Pair with None
50:5225.667 Send #2:5225.667 Receive #2:Null #3:Null Result:5225.667 Pair with None
51:5225.667 Send #2:5225.667 Receive #2:5225.667 #3:Null Result:5225.667 Pair with 2
52:5225.667 Send #2:5225.667 Receive #2:5225.667 #3:Null Result:5225.667 Pair with 2
53:5225.667 Send #2:5225.667 Receive #2:5225.667 #3:Null Result:5225.667 Pair with 2

Consensus agreement value 5225.666 is achieved at round 42.

REFERENCES

[1] C. C. Moallemi and B. Van Roy, “Consensus propagation,” *IEEE Trans. Inf. Theory*, vol. 52, no. 11, pp. 4753-4766, November 2006.

[2] R. Olfati-Saber, “Flocking for multi-agent dynamic system: Algorithms and theory,” *IEEE Trans. Autom. Control*, vol.51, no. 3, pp.401-420, March 2006

[3] R. Olfati-Saber, J. Alex Fax, and Richard M.murrray, “Consensus and cooperation in networked multi-agent systems,” *IEEE Proc.*, vol. 95, no. 1, pp. 215-233, January 2007

[4] R. Olfati-Saber and R. M. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp.1520-1533, sep.2004

[5] R. Olfati-Saber and R.M. Murray, “Consensus protocols for networks of dynamic agents,” in *Proc. 2003 Am. Control Conf.*, pp. 951-956, 2003.

[6] S. Buttery and A. Sago, “Future Applications of Bluetooth,” *BT Technology Journal*, vol. 21, no.3, pp. 48-55, 2003.

[7] C.D. Knutson, B. Young, E. Hall and D. Vawdrey, “Bluetooth wireless connectivity,” *IEEE Potentials*, vol. 21, no. 4, pp. 28-34, Dec 2002.

[8] P. K. Sahoo, C. Y. Chang and S. W. Chang, “Novel route maintenance protocols for the Bluetooth ad hoc network with mobility,” *Journal of Network and Computer Applications*, vol. 31, no. 4, pp. 535-558, November 2008.

[9] S. Avancha, A. Joshi, and T. Finin, “Enhanced Service Discovery in Bluetooth,” *IEEE Computer Society,* vol. 35, no. 6 pp. 96-99, Jun 2002.

[10] P. McDermott-Wells, “What is Bluetooth,” *IEEE Potentials*, vol. 23, no. 5, pp. 33-35, Dec 2004.

[11] K. Sairam, N. Gunasekaran and S.R. Redd, “Bluetooth in wireless communication,” *IEEE Communications Magazine*, vol. 40, no. 6, pp. 90-96, Jun 2002.

[12] A. Lanurie “Digital detective - Bluetooth,” *Digital Investigation*, vol. 3, no. 1, pp. 17-19, March 2006.

[13] J. Haartsen, M. Naghshineh, J. Inouye, O. J. Joeressen and W. Allen, “Bluetooth: vision, goals, and architecture,” *ACM SIGMOBILE Mobile Computing and Communications Review,* vol. 2, no. 4, pp. 38-45, October 1998.

[14] S. Sarkar, F. Anjum, and R. Guha, “Optimal Communication in Bluetooth Piconets,” *IEEE Vehicular Technology*, vol. 54, no. 2, pp. 709-721, Apr 2005.

[15] C. K. Toh, “Ad hoc mobile wireless networks: protocols and systems,” *Prentice Hall*, December, 2001

[16] T. D. Ndie, C. Tangha, T. Sangbong and A. F. Kufor, “Mobile Applications Provisioning Using Bluetooth,” *Journal of Software Engineering and Applications*, vol. 4, no. 2, pp. 95-105, 2011

[17] C. Metz, “Point to Point Protocol,” *IEEE Internet Computing*, vol. 3, no. 4, pp. 85-88, Aug 1999.

[18] S. Bocking, “Sockets++ a uniform application programming interface for basic level communication services,” *IEEE Communications Magazine*, vol. 34, no. 12, pp. 114-123, Aug 2002.

[19] A. Jones and J. Ohlund, “Network Programming for Microsoft Windows,” *Microsoft Press*, Feb 2002.

[20] M.K. Maggs, S.G. O'Keefe and D.V. Thiel, “Consensus Clock Synchronization for Wireless Sensor Networks” *IEEE Sensors Journal*, vol. 12, no. 6, pp. 2269-2277, Jun 2012.