

Santa Clara University
DEPARTMENT of COMPUTER ENGINEERING

Date: June 9, 2005

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY
SUPERVISION BY

Andry Winata

ENTITLED

Atomics: Ad Hoc Clock Synchronization for Handheld Devices

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER ENGINEERING

THESIS ADVISOR

DEPARTMENTCHAIR

**ATOMICS: AD HOC CLOCK SYNCHRONIZATION FOR
HANDHELD DEVICES**

by

Andry Winata

SENIOR DESIGN PROJECT REPORT

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Engineering
School of Engineering
Santa Clara University

Santa Clara, California

June 9, 2005

Atomics: Ad Hoc Clock Synchronization for Handheld Devices

Andry Winata

Department of Computer Engineering

Santa Clara University

2005

ABSTRACT

Clocks in handheld devices, such as Pocket PC's, do not perform well and are generally inaccurate. As is the case with their desktop counterparts, the clocks may lose or gain several seconds everyday, called a clock drift. This problem is inherent to the characteristics of hardware and software clocks used in a computing device, which can be alleviated is by regular synchronization to an Internet Time Server, such as the NIST Atomic Time Server. When access to the Internet is not available, it is often required to synchronize several devices to another device acting as the clock server. This project attempts to create a lightweight tool to reliably and accurately synchronize a handheld's system clock to an Internet Time Server, as well as to another handheld by means of an ad-hoc or temporary WiFi (802.11b / WLAN) connection. The tool also features an Automatic Server Discovery protocol which can detect the presence of a time server that broadcasts time information, instead of manually entering the server's name or IP address. Further development will enable the tool to work over other media such as Bluetooth and infrared, and to improve the accuracy and precision of the time synchronization process.

Keywords: Clock Synchronization, Handheld Devices, Ad-Hoc WiFi, Automatic Discovery, Atomic Time Server.

Acknowledgments

This thesis is dedicated to my parents, for their constant support and unconditional love.

I would like to thank Dr. Hans Peter Dommel, my senior design advisor, who helped me start this project and gave me valuable inputs and feedback.

I would also like to thank all of my friends who have always believed in me.

Table of Contents

SENIOR DESIGN PROJECT REPORT	2
Chapter 1: Introduction	6
1.1 Clocks in Computer Systems	7
1.2 The Need for Accurate Time Keeping	7
1.3 Accurate Clock in Handheld Device	8
1.4 Synchronization to an Atomic Time Server	9
1.5 Peer-to-Peer Handheld Synchronization	10
Chapter 2: Atomic Clock Synchronization Tool	11
2.1 Introduction	11
2.2 Internet Time Server Synchronization	11
2.3 Ad Hoc Wireless Synchronization	12
Chapter 3: Internet Time Server Synchronization	14
3.1 Internet Time Server	14
3.2 Daytime Protocol	15
3.3 Network Delay	16
3.4 Automatic Synchronization	17
<i>Figure 3.2: Automatic Synchronization Option</i>	17
Chapter 4: Ad Hoc Wireless Synchronization	18
4.1 Introduction	18
4.2 Wireless 802.11b (WiFi)	18
Chapter 5: Development	20
5.1 Hardware: Pocket PC 2002/2003 Handhelds	20
5.2 Software: Microsoft Visual Basic.NET	21
Chapter 6: Ethical Issues	23
References	24
Appendices (Source Code)	25

List of Figures

Figure 2.1 Synchronization over various Internet connections.....	12
Figure 2.2 Ad-Hoc Wireless Connection.....	13
Figure 3.1 Drop down list of servers	14
Figure 3.2 Automatic Synchronization Option.....	17
Figure 4.1 Server Discovery.....	18
Figure 4.2 Server Detected.....	19

Chapter 1: Introduction

1.1 Clocks in Computer Systems

A common misconception that many people have about the clock in their computer is how accurate it actually is. It is a *computer* clock, after all, so the clock must keep time much better than a cheap wrist-watch does. But the truth is that the wrist-watch is actually the better timekeeper. A computer clock can gain or lose anywhere from several seconds up to a full minute or more everyday. This is called a clock drift. This drift is inherent to the software and hardware clocks that computers use to keep time.

1.2 The Need for Accurate Time Keeping

The poor performance of computer clocks can cause many problems, because certain computer applications require time to be kept to the nearest seconds or better. For example, the computers in financial institutions must keep very accurate records of when transactions were completed, for legal or other reasons. In military institution, time-sensitive, mission-critical tasks require the use of highly accurate clocks. Accurate clocks are also essential for precise measurements in research and experiments. Software used on a manufacturing floor may need to turn a piece of equipment on or off at a specified time. Also, any system involved with synchronous communications must keep accurate time. For example, radio and television stations require both precise time-of-day and frequency in order to broadcast programs.

1.3 Accurate Clock in Handheld Device

Handheld devices were first developed by Apple with its Newton line of product. However, it was not until Palm released their handhelds with PalmOS that they become very popular. By focusing their software on personal information manager (PIM) applications, Palm succeeded to market their products to a wide range of users, from students to business executives. These PIM applications help the users to manage their schedule, keep a list of their contacts along with their extensive information, and keep track of their daily tasks. That is why handheld devices like the Palm are now called the PDA's (Personal Digital Assistants).

Microsoft, the largest software company in the world, also created its version of operating system for handheld devices, called the Windows CE. This operating system was used by handheld devices made by companies such as Hewlett-Packard with its Jornada line of product. Although Windows CE offered more features and capabilities to the device, many people preferred the PalmOS due to its simplicity and reliability. The battery life of the device is also an important factor since a Palm device can go for a week or longer while a Windows CE handheld can only last several hours on a battery.

However, the trend is currently changing. With the latest development in software and hardware technology, current handheld devices have faster processors and more memory that they are now able to run processing-intensive applications such as financial projections, movie player, and even 3D games. Many of today's handhelds have both WiFi and Bluetooth capabilities built-in that enable them to network wirelessly with other devices. Although battery life is still a limiting factor, it is a drawback that people seem to be willing to accept. The latest version Microsoft's operating system for mobile devices, Pocket PC 2003, can do just about any tasks a full desktop computer operating system can do. Due to the portability and the advancement in the hardware and software, people have begun using handheld devices for tasks that would normally require the use of a desktop or laptop computer. Although laptop computers will always have their own

uses, in many situations handhelds are more suitable and sometimes they are the only viable option.

As handhelds become more and more advanced, people will start using them for mission-critical applications that require accurate timing. Unfortunately, just like a desktop computer, a handheld device also suffers from clock drifts. This is the reason why an accurate time keeping is also needed on handheld devices.

1.4 Synchronization to an Atomic Time Server

One way to reduce the poor performance of computer clocks is to synchronize the clock to an atomic time server over the Internet on a regular basis. Atomic time servers – such as the National Institute of Standard and Technology (NIST) Internet Time Server – can be freely accessed by any computer connected to the Internet and provide an accurate time and date information. The server listens for timing information requests and when it receives one, it replies by sending the time back to the requesting party in the appropriate format. In order to do this synchronization, client software is needed to be run in the computer.

Internet time servers use several standard timing protocols defined in a series of RFC (Request for Comments) documents. The three major timing protocols are the Time Protocol, the Daytime Protocol, and the Network Time Protocol (NTP). Each protocol sends and receives the time information in different formats. They are also assigned to different port numbers, as the following describes:

- Time Protocol: Unformatted 32-bit binary number that contains time in UTC seconds since January 1, 1900. Uses port 37, tcp/ip, udp/ip.
- Daytime Protocol: Time code is sent as standard ASCII characters. Uses port 13, tcp/ip, udp/ip.

- Network Time Protocol (NTP): The server provides a data packet that includes a 64-bit timestamp containing the time in UTC seconds since January 1, 1900 with a resolution of 200 picoseconds. Uses port 123, udp/ip.
- Simple Network Time Protocol (SNTP): The data packet sent by the server is the same as NTP, but the client software does less processing and provides less accuracy. Uses port 123, udp/ip.

1.5 Peer-to-Peer Handheld Synchronization

In some cases, a user may want to synchronize a number of handhelds to a specific time. For example, a team of scientist making time measurements with their handhelds may need to set the clock on their handhelds precisely to a standard time. This can be achieved by setting a handheld to be a time server for the other handhelds. With the Ad-Hoc (Peer to Peer) capability of the 802.11b wireless network technology, the handhelds can directly connect to server handheld to get the time information without needing a centralized point of access (an Access Point).

Chapter 2: Atomics Clock Synchronization Tool

2.1 Introduction

The first part of the Atomics Clock Synchronization tool is designed to enable clock synchronization to Internet Time Servers. The second part of it will enable a handheld to act as a time server so that other handhelds may synchronize to it over any TCP/IP network, including an 802.11b wireless network in both Ad-Hoc (Peer to Peer) or Access Point (Centralized) mode.

2.2 Internet Time Server Synchronization

Using the Internet Time Server Synchronization feature, the users will have the ability to synchronize the clock whenever an Internet connection is available. The connection can be made through a wired LAN, wireless LAN, dial-up/PPP, or through the computer that the handheld is connected to. They will also be able configure the application to get the time information periodically, so they will not have to manually synchronize the clock everyday.

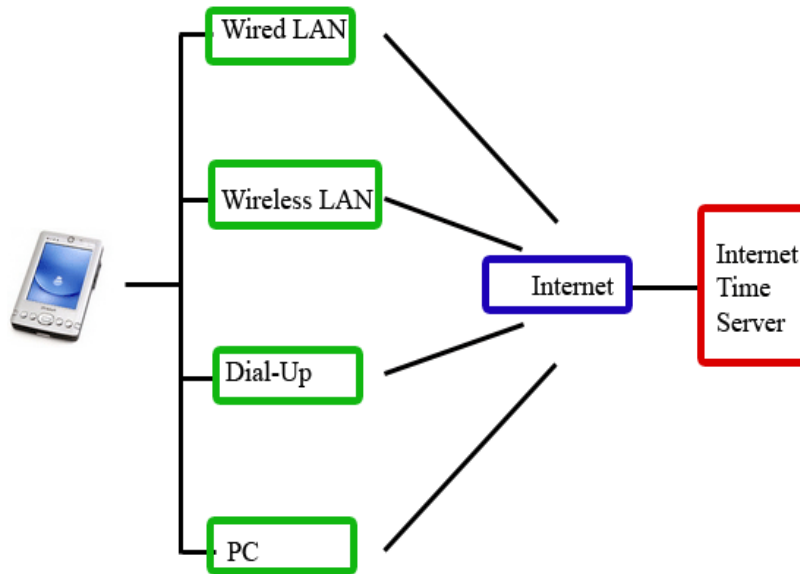


Figure 2.1: Synchronization can be done over various Internet connections

Several time servers will be displayed and the user can choose to connect to any one of them, or they can also enter their own preferred servers. Once they received a time information from the server, the time will be parsed according to the formats and be adjusted to the local time zone on the handheld. Once it is done, the handheld's current time will automatically be replaced with the new time.

2.3 Ad Hoc Wireless Synchronization

The user can also choose to set a handheld to be a time server for the other handheld. When this feature is enabled, the application will start the *listening* mode and wait for a *connection request* from other devices. The client handhelds can then send a request to connect in two ways: using the infra red connectivity (IRDA) or through a TCP/IP network. To make an infra red connection, both client and server handhelds need have their infra red ports align. They also need to have a clear line of sight (LOS) between

them, which means that they cannot have any objects in between. Once this is done, the client can simply make the request to the specified host name or ip address and the port number. The server will then acknowledge the request and reply with the time and date information.

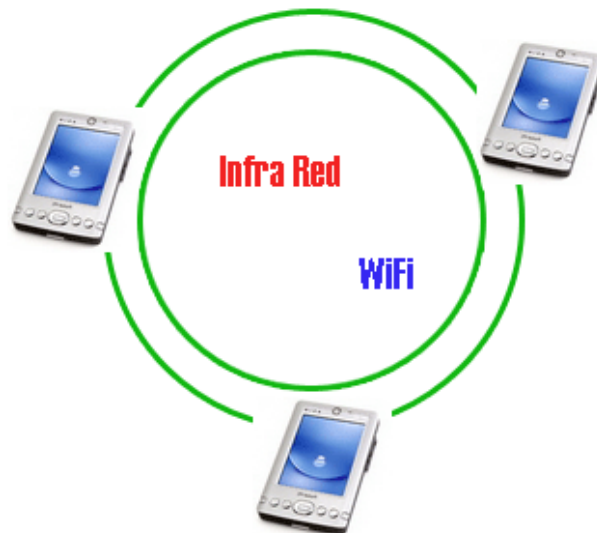


Figure 2.2: Ad-Hoc Wireless connection allow the devices to synchronize quickly

To make a connection over an Ad-Hoc (Peer to Peer) wireless LAN, the server and client handhelds simply have to be in the same ad-hoc wireless network. The Automatic Discovery capability will allow the client to seamlessly synchronize to the server handheld without needing to know its host name or ip address. When the server module is running, the handheld will advertise time information to every device in the network. The client will be able to detect the advertisement and prompt the user if he/she wants to synchronize to that time stamp.

Chapter 3: Internet Time Server Synchronization

3.1 Internet Time Server

As was discussed previously, many network time servers are freely accessible over the Internet, such as the National Institute of Standard and Technology (NIST) Atomic Time Server. The Atomics Clock Synchronization tool is designed to synchronize with any Internet Time Server that uses the Daytime protocol. The user may choose one of the servers listed in the drop down list, or enter their own preferred server.

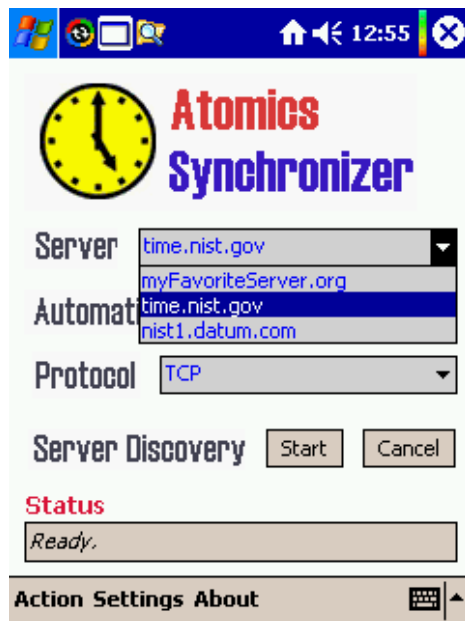


Figure 3.1: Drop down list of servers

3.2 Daytime Protocol

This network time/date protocol was chosen because of its simplicity, which complements the limited system calls on handheld devices. The daytime protocol operates on port 13 with the time code sent as standard ASCII characters. It can use either TCP/IP or UDP/IP stack. The standard is defined in RFC-867. Although the format may vary from server to server, it mostly follows this form:

JJJJ YR-MO-DA HH:MM:SS TT L H msADV UTC(NIST) OTM

where:

- JJJJ is the Modified Julian Date (MJD). The MJD is the last five digits of the Julian Date, which is simply a count of the number of days since January 1, 4713 B.C. To get the Julian Date, add 2.4 million to the MJD.
- YR-MO-DA is the date. It shows the last two digits of the year, the month, and the current day of month.
- HH:MM:SS is the time in hours, minutes, and seconds. The time is always sent as Coordinated Universal Time (UTC). An offset needs to be applied to UTC to obtain local time.
- TT is a two digit code (00 to 99) that indicates whether the United States is on Standard Time (ST) or Daylight Saving Time (DST). It also indicates when ST or DST is approaching. This code is set to 00 when ST is in effect, or to 50 when DST is in effect. During the month in which the time change actually occurs, this number will decrement every day until the change occurs. For example, during the month of October, the U.S. changes from DST to ST.
- L is a one-digit code that indicates whether a leap second will be added or subtracted at midnight on the last day of the current month. If the code is 0, no leap second will occur this month. If the code is 1, a positive leap second will be

added at the end of the month. This means that the last minute of the month will contain 61 seconds instead of 60. If the code is 2, a second will be deleted on the last day of the month. Leap seconds occur at a rate of about one per year.

- H is a health digit that indicates the health of the server. If H=0, the server is healthy. If H=1, then the server is operating properly but its time may be in error by up to 5 seconds. This state should change to fully healthy within 10 minutes. If H=2, then the server is operating properly but its time is known to be wrong by more than 5 seconds. If H=4, then a hardware or software failure has occurred and the amount of the time error is unknown.
- msADV displays the number of milliseconds that NIST advances the time code to partially compensate for network delays. The advance is currently set to 50.0 milliseconds.
- OTM (on-time marker) is an asterisk (*). The time values sent by the time code refer to the arrival time of the OTM. In other words, if the time code says it is 12:45:45, this means it is 12:45:45 when the OTM arrives.

3.3 Network Delay

During the synchronization, the time that it takes for the time stamp to travel from the server to the client is called the server-client propagation delay. We need to measure the server-client propagation delay to determine its clock offset relative. While it is impossible to obtain the actual length of the delay, we can approximate it using the following formula:

$$\text{offset} = [(T2 - T1) + (T3 - T4)] / 2$$

$$\text{delay} = (T4 - T1) - (T3 - T2) \text{ (Roundtrip)}$$

T1 = client timestamp on the request message
T2 = server timestamp upon arrival
T3 = the server timestamp on departure of reply
T4 = the client timestamp upon arrival.

With this formula, we know that the margin of error is always less than half of the roundtrip delay. This is useful when we want to measure the correctness of the time information.

3.4 Automatic Synchronization

The user may choose to automatically synchronize the clock on a certain interval. The user can choose an interval of 15 minutes, one hour, or one day. Simply closing (not exiting) the program will send it running in the background and automatically synchronize to the server that was chosen.

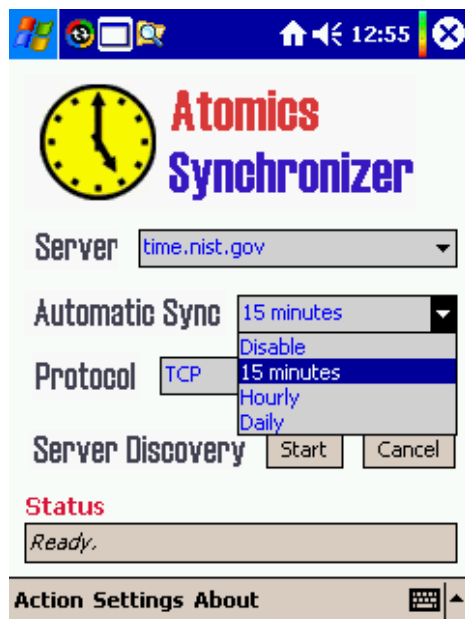


Figure 3.2: Automatic Synchronization Option

Chapter 4: Ad Hoc Wireless Synchronization

4.1 Introduction

When access to the Internet is not available, an immediate, on-the go synchronization is sometimes needed for special purposes, such as synchronizing the clock of several handhelds to a pre-determined, arbitrary time. In this case, one handheld can become a mobile time server and synchronize its time stamp to the other handhelds.

4.2 Wireless 802.11b (WiFi)

The 802.11b has become the most popular wireless technology in the mobile computing world for the past several years. Most of the latest handhelds now include WiFi capability out of the box. It has a range of over 30 meters and does not require a line of sight. These are the main reasons for my decision to develop the tool to work over WiFi. However, future development can also include infrared and Bluetooth media.

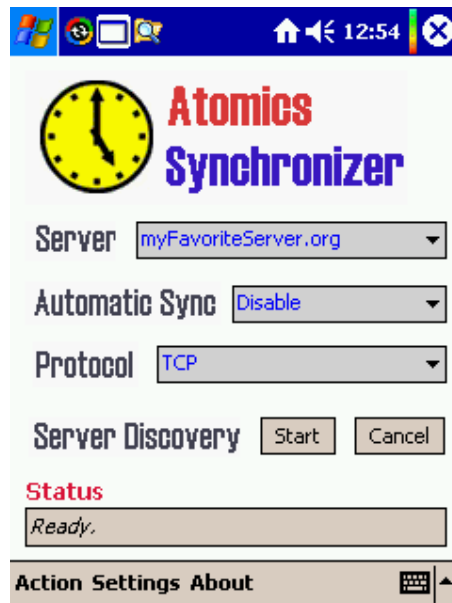


Figure 4.1: Server Discovery

4.3 Automatic Server Discovery

When operating in the wireless Ad-Hoc mode, handhelds can automatically detect the presence of a time server using the Automatic Server Discovery feature. This feature allows a server handheld to broadcast UDP packets to advertise its presence. The client handheld listens to this broadcast and asks the user if he/she would like to synchronize to that particular server. This feature eliminates the need for the client to manually enter the machine name or the IP address of the server, while at the same time accommodates synchronizing to multiple client handhelds simultaneously.

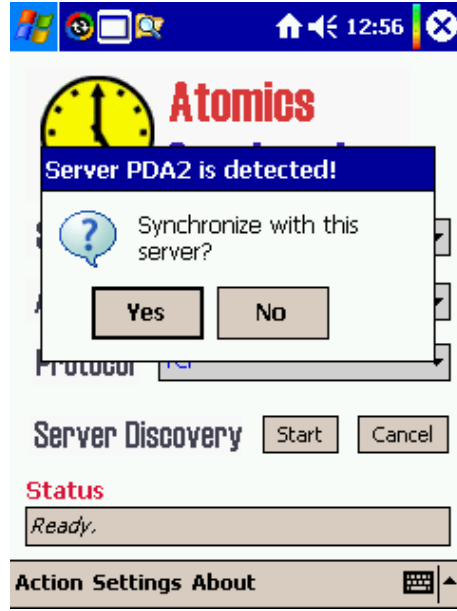


Figure 4.2: Server Detected

Chapter 5: Development

5.1 Hardware: Pocket PC 2002/2003 Handhelds

The target devices are handheld devices running Microsoft Pocket PC 2002 or 2003. There are several reasons why I choose Pocket PC devices over Palm devices as the target device. Most handheld users connect to the Internet and to other devices by means of wireless (WiFi) connection. Many Pocket PC handhelds now have integrated wireless network card, while only a small number of Palm devices have it. Another reason is because that the number of Pocket PC users is steadily growing larger than Palm users. More manufacturers put Pocket PC operating system in their handhelds and cell phones, such as HP, Toshiba, Dell, O2, and Asus, while Sony – a big player in Palm world with their *Clie* product line – has already quit the handheld industry.

The latest Pocket PC devices also have more powerful hardware and new features such as integrated Bluetooth, VGA screen for higher resolution, multimedia accelerator with video memory, built-in digital camera, and fast 624MHz processors. In short, I foresee Pocket PC operating system, software, and hardware continues to grow rapidly over the next several years.



Figure 5.1: Dell Axim X30 with built-in Bluetooth and WiFi capabilities

5.2 Software: Microsoft Visual Basic.NET

Language: Microsoft Visual Basic.NET

Environment: Microsoft Visual Studio.NET

This programming language was chosen for several reasons:

- Allows rapid development, which is important due to the time constraint of this project
- Uses Microsoft CF.NET (Compact Framework), which was designed for mobile devices.
- My previous experience with eVB and Visual Basic 6.0.
- Cost effective at \$100 for the academic version.

5.3 External Components

Additional software tool: The trial version of IP*Works Networking Tool.

- It was used only in the UDP broadcasting part

5.4 Major Challenges

Handheld programming has not been around for long and therefore the resources were limited. I could only find a handful of books on the subject and online research seldom resulted in relevant and up-to-date information. This alone was the major challenge in programming for handheld devices.

Wireless networking is not as reliable as wired networking. Sometimes signal would drop randomly or the packets would not get through. It required many trial and errors and it got tedious and frustrating at times. Handling asynchronous communication also means that it is event-based which makes it harder to debug, because the program does not run in a linear and predictable way.

Cost was also an issue, as I had to purchase two high end Pocket PC handhelds to develop this project.

Another issue is definitely the time constraint in which we have to design, develop, present, and document this project.

5.5 Development Process:

The entire project was developed using an iterative prototype process, by building one component at a time and then implementing it into the final version.

Chapter 6: Ethical Issues

This software tool is very limited and specific to one part of computing that does not relate to social, ethical, political, economical, or health issues in a significant way.

Sustainability: A clock synchronization tool will always be needed as long as the design of clocks in computing devices does not change.

Usability: The program is straightforward and easy to use. It has definite and very practical purposes.

Lifelong learning: This project helped prepare me to learn on my own and inspired me to study new subjects. I realized that many things cannot be found in books and I needed to be creative.

References

1. Makofsky, Steve, *Pocket PC Network Programming*, San Francisco: Addison Wesley, 2004.
2. Tacke Christopher, and Basset Tim, *eMbedded Visual Basic – Windows CE and Pocket PC Mobile Application*, Indianapolis IN: Sams, 2001.
3. Grattan N., Pocket PC, *Handheld PC – Developer’s Guide*, NJ: Prentice Hall, 2002.
4. National Institute of Standards and Technology Website
<http://nist.gov/>
5. Microsoft Mobile and Embedded Developer Center Website
<http://msdn.microsoft.com/mobility/>

Appendices

File List

1. AssemblyInfo.vb
2. AtomicSync.resx
3. AtomicSync.vb
4. GUI.exe
5. GUI.sln
6. GUI.vbdproj
7. GUI.vbdproj.user
8. logo.bmp
9. Microsoft.WindowsCE.Forms.dll
10. Microsoft.WindowsCE.Forms.xml
11. mscorlib.dll
12. mscorlib.xml
13. Registry.vb

Source Code: AtomicSync.vb

```
Imports System
Imports System.Text
Imports System.IO
Imports System.Net
Imports System.Net.Sockets
Imports System.Text.RegularExpressions
Imports System.Runtime.InteropServices

Public Class Form1
    Inherits System.Windows.Forms.Form

    'DECLARE THE GUI AND MENU CLASSES
    Friend WithEvents PictureBox1 As System.Windows.Forms.PictureBox
    Friend WithEvents MainMenu1 As System.Windows.Forms.MainMenu

    'DECLARE THE IPWORKS SOCKET COMPONENTS
    Protected WithEvents UdpSocket1 As New nsoftware.IPWorks.Udpport
    Protected WithEvents UdpSocket2 As New nsoftware.IPWorks.Udpport
    Protected WithEvents TcpSocket1 As New nsoftware.IPWorks.Ipport

    'DECLARE GLOBAL VARIABLES
    Private receivedData As String
    Private newFormattedDate As String
    Private GMTOffset As Integer
    Private closeServer As Boolean
    Private protocolType As String
    Private isPeer As Boolean
    Private syncInterval As Integer
    Private timerMsg As String

    'DECLARE THE SYSTEMTIME STRUCTURE FOR SETTING LOCAL TIME
    Public Structure SYSTEMTIME
        Public year As Short
        Public month As Short
        Public dayOfWeek As Short
        Public day As Short
        Public hour As Short
        Public minute As Short
        Public second As Short
        Public milliseconds As Short
    End Structure

    'DECLARE THE SETLOCALTIME API FUNCTION
    <DllImport("coredll.dll")> Private Shared Function
    SetLocalTime(ByRef time As SYSTEMTIME) As Boolean
    End Function

#Region " Windows Form Designer generated code "

    Public Sub New()
```

```

MyBase.New()

'This call is required by the Windows Form Designer.
InitializeComponent()

'Add any initialization after the InitializeComponent() call

End Sub

'Form overrides dispose to clean up the component list.
Protected Overloads Overrides Sub Dispose(ByVal disposing As
Boolean)
    MyBase.Dispose(disposing)
End Sub

'NOTE: The following procedure is required by the Windows Form
Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
Friend WithEvents Label4 As System.Windows.Forms.Label
Friend WithEvents Button1 As System.Windows.Forms.Button
Friend WithEvents TextBox1 As System.Windows.Forms.TextBox
Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem2 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem3 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem4 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem5 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem6 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem7 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem8 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem9 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem10 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem11 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem12 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem13 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem14 As System.Windows.Forms.MenuItem
Friend WithEvents MenuItem15 As System.Windows.Forms.MenuItem
Friend WithEvents ComboBox1 As System.Windows.Forms.ComboBox
Friend WithEvents ComboBox2 As System.Windows.Forms.ComboBox
Friend WithEvents ComboBox3 As System.Windows.Forms.ComboBox
Friend WithEvents PictureBox2 As System.Windows.Forms.PictureBox
Friend WithEvents PictureBox3 As System.Windows.Forms.PictureBox
Friend WithEvents PictureBox4 As System.Windows.Forms.PictureBox
Friend WithEvents PictureBox5 As System.Windows.Forms.PictureBox
Friend WithEvents Button2 As System.Windows.Forms.Button
Friend WithEvents Timer1 As System.Windows.Forms.Timer
Friend WithEvents BroadcastTimer As System.Windows.Forms.Timer
Friend WithEvents MenuItem16 As System.Windows.Forms.MenuItem
Private Sub InitializeComponent()
    Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(Form1))
    Me.MainMenu1 = New System.Windows.Forms.MainMenu
    Me.MenuItem1 = New System.Windows.Forms.MenuItem
    Me.MenuItem2 = New System.Windows.Forms.MenuItem
    Me.MenuItem16 = New System.Windows.Forms.MenuItem
    Me.MenuItem4 = New System.Windows.Forms.MenuItem
    Me.MenuItem3 = New System.Windows.Forms.MenuItem

```

```

Me.MenuItem5 = New System.Windows.Forms.MenuItem
Me.MenuItem13 = New System.Windows.Forms.MenuItem
Me.MenuItem6 = New System.Windows.Forms.MenuItem
Me.MenuItem7 = New System.Windows.Forms.MenuItem
Me.MenuItem14 = New System.Windows.Forms.MenuItem
Me.MenuItem8 = New System.Windows.Forms.MenuItem
Me.MenuItem9 = New System.Windows.Forms.MenuItem
Me.MenuItem10 = New System.Windows.Forms.MenuItem
Me.MenuItem11 = New System.Windows.Forms.MenuItem
Me.MenuItem12 = New System.Windows.Forms.MenuItem
Me.MenuItem15 = New System.Windows.Forms.MenuItem
Me.PictureBox1 = New System.Windows.Forms.PictureBox
Me.Label4 = New System.Windows.Forms.Label
Me.Button1 = New System.Windows.Forms.Button
Me.TextBox1 = New System.Windows.Forms.TextBox
Me.ComboBox1 = New System.Windows.Forms.ComboBox
Me.ComboBox2 = New System.Windows.Forms.ComboBox
Me.ComboBox3 = New System.Windows.Forms.ComboBox
Me.PictureBox2 = New System.Windows.Forms.PictureBox
Me.PictureBox3 = New System.Windows.Forms.PictureBox
Me.PictureBox4 = New System.Windows.Forms.PictureBox
Me.PictureBox5 = New System.Windows.Forms.PictureBox
Me.Button2 = New System.Windows.Forms.Button
Me.Timer1 = New System.Windows.Forms.Timer
Me.BroadcastTimer = New System.Windows.Forms.Timer
'
'MainMenu1
'
Me.MainMenu1.MenuItems.Add(Me.MenuItem1)
Me.MainMenu1.MenuItems.Add(Me.MenuItem6)
Me.MainMenu1.MenuItems.Add(Me.MenuItem8)
'
'MenuItem1
'
Me.MenuItem1.MenuItems.Add(Me.MenuItem2)
Me.MenuItem1.MenuItems.Add(Me.MenuItem16)
Me.MenuItem1.MenuItems.Add(Me.MenuItem4)
Me.MenuItem1.MenuItems.Add(Me.MenuItem3)
Me.MenuItem1.MenuItems.Add(Me.MenuItem5)
Me.MenuItem1.MenuItems.Add(Me.MenuItem13)
Me.MenuItem1.Text = "Action"
'
'MenuItem2
'
Me.MenuItem2.Text = "Start Sync"
'
'MenuItem16
'
Me.MenuItem16.Enabled = False
Me.MenuItem16.Text = "Stop Sync"
'
'MenuItem4
'
Me.MenuItem4.Text = "-"
'
'MenuItem3
'

```

```

Me.MenuItem3.Text = "Run Server"
'
'MenuItem5
'
Me.MenuItem5.Enabled = False
Me.MenuItem5.Text = "Stop Server"
'
'MenuItem13
'
Me.MenuItem13.Text = "-"
'
'MenuItem6
'
Me.MenuItem6.MenuItems.Add(Me.MenuItem7)
Me.MenuItem6.MenuItems.Add(Me.MenuItem14)
Me.MenuItem6.Text = "Settings"
'
'MenuItem7
'
Me.MenuItem7.Text = "Save"
'
'MenuItem14
'
Me.MenuItem14.Text = "-"
'
'MenuItem8
'
Me.MenuItem8.MenuItems.Add(Me.MenuItem9)
Me.MenuItem8.MenuItems.Add(Me.MenuItem10)
Me.MenuItem8.MenuItems.Add(Me.MenuItem11)
Me.MenuItem8.MenuItems.Add(Me.MenuItem12)
Me.MenuItem8.MenuItems.Add(Me.MenuItem15)
Me.MenuItem8.Text = "About"
'
'MenuItem9
'
Me.MenuItem9.Text = "About"
'
'MenuItem10
'
Me.MenuItem10.Text = "Help"
'
'MenuItem11
'
Me.MenuItem11.Text = "-"
'
'MenuItem12
'
Me.MenuItem12.Text = "Exit"
'
'MenuItem15
'
Me.MenuItem15.Text = "-"
'
'PictureBox1
'

```

```

        Me.PictureBox1.Image =
CType(resources.GetObject("PictureBox1.Image"), System.Drawing.Image)
        Me.PictureBox1.Location = New System.Drawing.Point(8, 8)
        Me.PictureBox1.Size = New System.Drawing.Size(224, 80)
        '
        'Label4
        '
        Me.Label4.Font = New System.Drawing.Font("Tahoma", 9.0!,
System.Drawing.FontStyle.Bold)
        Me.Label4.ForeColor = System.Drawing.Color.Crimson
        Me.Label4.Location = New System.Drawing.Point(8, 224)
        Me.Label4.Size = New System.Drawing.Size(100, 16)
        Me.Label4.Text = "Status"
        '
        'Button1
        '
        Me.Button1.Font = New System.Drawing.Font("Tahoma", 8.25!,
System.Drawing.FontStyle.Regular)
        Me.Button1.Location = New System.Drawing.Point(133, 192)
        Me.Button1.Size = New System.Drawing.Size(40, 20)
        Me.Button1.Text = "Start"
        '
        'TextBox1
        '
        Me.TextBox1.BackColor = System.Drawing.SystemColors.Control
        Me.TextBox1.Font = New System.Drawing.Font("Tahoma", 8.25!,
System.Drawing.FontStyle.Italic)
        Me.TextBox1.ForeColor =
System.Drawing.SystemColors.ControlDarkDark
        Me.TextBox1.Location = New System.Drawing.Point(8, 240)
        Me.TextBox1.Size = New System.Drawing.Size(224, 21)
        Me.TextBox1.Text = "TextBox1"
        '
        'ComboBox1
        '
        Me.ComboBox1.Font = New System.Drawing.Font("Haettenschweiler",
8.25!, System.Drawing.FontStyle.Regular)
        Me.ComboBox1.Location = New System.Drawing.Point(67, 88)
        Me.ComboBox1.Size = New System.Drawing.Size(165, 20)
        '
        'ComboBox2
        '
        Me.ComboBox2.Font = New System.Drawing.Font("Tahoma", 8.25!,
System.Drawing.FontStyle.Regular)
        Me.ComboBox2.Location = New System.Drawing.Point(118, 122)
        Me.ComboBox2.Size = New System.Drawing.Size(114, 21)
        '
        'ComboBox3
        '
        Me.ComboBox3.Font = New System.Drawing.Font("Tahoma", 8.25!,
System.Drawing.FontStyle.Regular)
        Me.ComboBox3.Location = New System.Drawing.Point(78, 154)
        Me.ComboBox3.Size = New System.Drawing.Size(154, 21)
        '
        'PictureBox2
        '

```

```

        Me.PictureBox2.Image =
CType(resources.GetObject("PictureBox2.Image"), System.Drawing.Image)
        Me.PictureBox2.Location = New System.Drawing.Point(8, 88)
        Me.PictureBox2.Size = New System.Drawing.Size(48, 24)
        '
        'PictureBox3
        '
        Me.PictureBox3.Image =
CType(resources.GetObject("PictureBox3.Image"), System.Drawing.Image)
        Me.PictureBox3.Location = New System.Drawing.Point(8, 120)
        Me.PictureBox3.Size = New System.Drawing.Size(104, 24)
        '
        'PictureBox4
        '
        Me.PictureBox4.Image =
CType(resources.GetObject("PictureBox4.Image"), System.Drawing.Image)
        Me.PictureBox4.Location = New System.Drawing.Point(12, 152)
        Me.PictureBox4.Size = New System.Drawing.Size(64, 24)
        '
        'PictureBox5
        '
        Me.PictureBox5.Image =
CType(resources.GetObject("PictureBox5.Image"), System.Drawing.Image)
        Me.PictureBox5.Location = New System.Drawing.Point(10, 192)
        Me.PictureBox5.Size = New System.Drawing.Size(112, 24)
        '
        'Button2
        '
        Me.Button2.Enabled = False
        Me.Button2.Font = New System.Drawing.Font("Tahoma", 8.25!,
System.Drawing.FontStyle.Regular)
        Me.Button2.Location = New System.Drawing.Point(183, 192)
        Me.Button2.Size = New System.Drawing.Size(48, 20)
        Me.Button2.Text = "Cancel"
        '
        'Timer1
        '
        'BroadcastTimer
        '
        'Form1
        '
        Me.Controls.Add(Me.Button2)
        Me.Controls.Add(Me.PictureBox5)
        Me.Controls.Add(Me.PictureBox4)
        Me.Controls.Add(Me.PictureBox3)
        Me.Controls.Add(Me.PictureBox2)
        Me.Controls.Add(Me.ComboBox3)
        Me.Controls.Add(Me.ComboBox2)
        Me.Controls.Add(Me.ComboBox1)
        Me.Controls.Add(Me.TextBox1)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.Label4)
        Me.Controls.Add(Me.PictureBox1)
        Me.Menu = Me.MainMenu1
        Me.Text = "Atomic Soft"

```

```

End Sub

#End Region

'SETTING UP THE GUI PART AND THE COLORS
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    TextBox1.BackColor = System.Drawing.Color.Gray
    Progress("Ready.")

    ComboBox1.Items.Add("time.nist.gov")
    ComboBox1.Items.Add("nist1.datum.com")
    ComboBox1.Items.Add("myFavoriteServer.org")
    ComboBox1.SelectedIndex = 0
    ComboBox1.BackColor = System.Drawing.Color.LightGray
    ComboBox1.ForeColor = System.Drawing.Color.Blue

    ComboBox2.Items.Add("Disable")
    ComboBox2.Items.Add("15 minutes")
    ComboBox2.Items.Add("Hourly")
    ComboBox2.Items.Add("Daily")
    ComboBox2.SelectedIndex = 0
    ComboBox2.BackColor = System.Drawing.Color.LightGray
    ComboBox2.ForeColor = System.Drawing.Color.Blue

    ComboBox3.Items.Add("TCP")
    ComboBox3.Items.Add("IRDA")
    ComboBox3.SelectedIndex = 0
    ComboBox3.BackColor = System.Drawing.Color.LightGray
    ComboBox3.ForeColor = System.Drawing.Color.Blue

End Sub

'STATUS/MESSAGE BAR
Private Sub Progress(ByVal Msg As String)
    TextBox1.Text = Msg
End Sub

'MENU ITEM: SYNCHRONIZE
'SYNCHRONIZATION WITH AN INTERNET TIME SERVER
Private Sub MenuItem2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MenuItem2.Click
    'DISABLE OTHER MENU ITEMS
    MenuItem2.Enabled = False
    MenuItem3.Enabled = False
    MenuItem5.Enabled = False

    'CHECK THE VALUE OF AUTOMATIC SYNC INTERVAL
    Select Case ComboBox2.Text

        Case "Disabled"
            syncInterval = 0

        Case "15 minutes"

```

```

        syncInterval = 15
        timerMsg = "every 15 minutes.."

    Case "Hourly"
        syncInterval = 60
        timerMsg = "every hour.."

    Case "Daily"
        syncInterval = 1440
        timerMsg = "once a day.."

End Select

'IF AUTOMATIC SYNC IS ENABLED, SETUP TIMER
'OTHERWISE JUST SYNCHRONIZE ONCE
If syncInterval > 0 Then
    'CONVERT FROM MINUTES TO MILLISECONDS
    Timer1.Interval = syncInterval * 60 * 1000
    Timer1.Enabled = True
    Progress("Automatic sync " & timerMsg)
Else
    ConnectToServer()
End If

End Sub

'CONNECTING TO THE SERVER
Private Sub ConnectToServer()
    Progress("Connecting to " & ComboBox1.Text & "..")

    'CHECK IF TCPSOCKET IS ALREADY CONNECTED
    If TcpSocket1.Connected = False Then
        TcpSocket1.RemoteHost = ComboBox1.Text
        TcpSocket1.RemotePort = 13
        TcpSocket1.LocalPort = 13
        TcpSocket1.Connected = True
    End If
End Sub

Private Sub TcpSocket1_OnDataIn(ByVal sender As Object, ByVal e As
nsoftware.IPWorks.IpportDataInEventArgs) Handles TcpSocket1.OnDataIn
    receivedData = e.Text
    Progress("Reply is: " & receivedData)
    Progress(TimeString)
    SetDateAndTime()
    TcpSocket1.Linger = False
    TcpSocket1.Connected = False
End Sub

'SETTING THE NEW DATE AND TIME INTO THE SYSTEM CLOCK
Private Sub SetDateAndTime()
    Progress("Starting Setting Date..")
    FormatNewDate(receivedData)
    Dim newTimeInfo As Date

```

```

newTimeInfo = newFormattedDate
Dim newTime As Date
Dim newDate As Date

If Not isPeer Then
    GetGMTOffset()
    Progress("GMT:" & GMTOffset)
    newTimeInfo = DateAdd("n", GMTOffset, newTimeInfo)
End If

'newTime = Trim(Microsoft.VisualBasic.Right(newTimeInfo, 11))

Dim aSystemTime As SYSTEMTIME
aSystemTime.year = newTimeInfo.Year
aSystemTime.month = newTimeInfo.Month
aSystemTime.dayOfWeek = newTimeInfo.DayOfWeek
aSystemTime.day = newTimeInfo.Day
aSystemTime.hour = newTimeInfo.Hour
aSystemTime.minute = newTimeInfo.Minute
aSystemTime.second = newTimeInfo.Second
aSystemTime.milliseconds = newTimeInfo.Millisecond

'Set the new time...
SetLocalTime(aSystemTime)

Progress("Successfully synchronized to: " & newTimeInfo)

End Sub

'GET THE GMT OFFSET FOR THE SYSTEM REGISTRY
Private Sub GetGMTOffset()
    Const RegPath As String = "SOFTWARE\Microsoft\Clock"

    GMTOffset = Registry.GetInt(Registry.RootKey.LocalMachine,
RegPath, "GMT_OFFSET")

    GMTOffset = -(GMTOffset)
    GMTOffset = GMTOffset + 60
End Sub

'CONVERTING THE RECEIVED INFORMATION TO THE CORRECT DATE AND TIME
FORMAT
Private Sub FormatNewDate(ByVal NetTime As String)
    Progress("Starting formatting...")
    Dim strDate As String
    Dim strTime As String
    Dim checkTag As String
    Dim yr As String
    Dim mn As String
    Dim dy As String

    checkTag = Trim(Mid(NetTime, 1, 6))

    'CHECK IF THIS IS A PEER-TO-PEER SYNCHRONIZATION
    If checkTag = "<peer>" Then

```

```

        isPeer = True
        strDate = Trim(Mid(NetTime, 7))

        If IsDate(strDate) Then
            newFormattedDate = strDate
        Else
            newFormattedDate = Now
        End If

    Else
        isPeer = False
        strDate = Trim(Mid(NetTime, 7, 9))

        'PARSE THE STRING DATA
        yr = Trim(Mid(strDate, 1, 2))
        mn = Trim(Mid(strDate, 4, 2))
        dy = Trim(Mid(strDate, 7, 2))

        strDate = mn & "-" & dy & "-" & yr

        strTime = Trim(Mid(NetTime, 16, 9))

        'IF DATA IS INVALID, SET SYSTEM CLOCK TO CURRENT TIME
        If IsDate(strDate) And IsDate(strTime) Then
            newFormattedDate = CDate(strDate & " " & strTime)
        Else
            newFormattedDate = Now
        End If

    End If

End Sub

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
    ConnectToServer()
End Sub

'MENU ITEM: RUN SERVER
Private Sub MenuItem3_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MenuItem3.Click
    'DISABLE OTHER MENU ITEMS
    MenuItem2.Enabled = False
    MenuItem3.Enabled = False
    MenuItem16.Enabled = False
    Button1.Enabled = False
    Button2.Enabled = False
    MenuItem5.Enabled = True

    InitializeBroadcast()

    'SEND OUT BROADCAST EVERY 5 SECONDS
    BroadcastTimer.Interval = 5000
    BroadcastTimer.Enabled = True
End Sub

```

```

Private Sub InitializeBroadcast()
    'TRY TO DEACTIVATE SOCKET IF IT'S ACTIVE
    If UdpSocket1.Active = True Then
        UdpSocket1.Active = False
    End If

    'ONLY ACTIVATE SOCKET WHEN IT'S UNACTIVE
    If UdpSocket1.Active = False Then
        Progress("Initializing broadcast..")
        UdpSocket1.RemoteHost = "255.255.255.255"
        UdpSocket1.RemotePort = 2255
        UdpSocket1.LocalPort = 2255
        UdpSocket1.Active = True
        UdpSocket1.AcceptData = False
    End If
End Sub

Private Sub BroadcastTimer_Tick(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles BroadcastTimer.Tick
    Dim currentTime = Now
    Progress("Broadcasting: " & currentTime)
    UdpSocket1.DataToSend = "<peer> " & currentTime
    'UdpSocket1.DataToSend = currentTime
End Sub

'MENU ITEM: STOP SERVER
Private Sub MenuItem5_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MenuItem5.Click

    MenuItem2.Enabled = True
    MenuItem3.Enabled = True
    Button1.Enabled = True
    Button2.Enabled = False
    MenuItem16.Enabled = False
    MenuItem5.Enabled = False

    'STOP TIMER ONLY IF IT'S RUNNING
    If BroadcastTimer.Enabled = True Then
        BroadcastTimer.Enabled = False
    End If

    'DEACTIVATE SOCKET ONLY IF IT'S ACTIVE
    If UdpSocket1.Active = True Then
        UdpSocket1.Active = False
    End If

    Progress("Broadcasting stopped..")

End Sub

'MENU ITEM: STOP SYNC - STOP AUTOMATIC SYNC
Private Sub MenuItem16_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MenuItem16.Click
    MenuItem2.Enabled = True
    MenuItem3.Enabled = True

```

```

MenuItem5.Enabled = True

'STOP TIMER
If Timer1.Enabled = True Then
    Timer1.Enabled = False
End If

'CLOSE SOCKET
If TcpSocket1.Connected = True Then
    TcpSocket1.Connected = False
End If

Progress("(Automatic) synchronization stopped..")
End Sub

'BUTTON: SERVER DISCOVERY - START
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    Button1.Enabled = False
    Button2.Enabled = True

    MenuItem2.Enabled = False
    MenuItem3.Enabled = False
    MenuItem5.Enabled = False
    MenuItem16.Enabled = False

    Progress("Initializing discovery..")
    InitializeDiscovery()
End Sub

Private Sub InitializeDiscovery()
    'TRY TO DEACTIVATE SOCKET IF IT'S ACTIVE
    If UdpSocket2.Active = True Then
        UdpSocket2.Active = False
    End If

    'ACTIVATE SOCKET ONLY WHEN IT'S UNACTIVE
    If UdpSocket2.Active = False Then
        UdpSocket2.RemotePort = 2255
        UdpSocket2.LocalPort = 2255
        UdpSocket2.Active = True
        UdpSocket2.AcceptData = True
        Progress("Detecting nearby server..")
    End If
End Sub

Private Sub UdpSocket2_OnDataIn(ByVal sender As Object, ByVal e As
nsoftware.IPWorks.UdpportDataInEventArgs) Handles UdpSocket2.OnDataIn
    Progress("Received: " & e.Datagram & " from " &
e.SourceAddress)
    receivedData = e.Datagram
    'ONCE DATA IS RECEIVED, PAUSE ACCEPTING MORE DATA
    UdpSocket2.AcceptData = False

    Dim Msg As String = "Synchronize with this server?"

```

```

        Dim Cap As String = "Server " & e.SourceAddress & " is
detected!"

        Dim Result As DialogResult

        Result = MessageBox.Show(Msg, Cap, MessageBoxButtons.YesNo,
MessageBoxIcon.Question, MessageBoxDefaultButton.Button1)

        If Result = DialogResult.Yes Then
            'Progress("We're here..")
            SetDateAndTime()
        Else
            Progress("Synchronization denied..")
        End If

    End Sub

    'BUTTON: SERVER DISCOVERY - CANCEL
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Button1.Enabled = True
        Button2.Enabled = False

        MenuItem2.Enabled = True
        MenuItem3.Enabled = True
        MenuItem5.Enabled = True
        MenuItem16.Enabled = True

        If UdpSocket2.AcceptData = True Then
            UdpSocket2.AcceptData = False
        End If

        If UdpSocket2.Active = True Then
            UdpSocket2.Active = False
        End If

        Progress("Discovery stopped..")
    End Sub

    'MENU ITEM: ABOUT
    Private Sub MenuItem9_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MenuItem9.Click

        Dim Msg As String = "Atomic Synchronizer version 1.0, by Andry
Winata."
        Dim Cap As String = "About"

        MessageBox.Show(Msg, Cap, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk, MessageBoxDefaultButton.Button1)

    End Sub

```

```
'MENU ITEM: HELP
Private Sub MenuItem10_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MenuItem10.Click

    Dim Msg As String = "For help on using this software, visit
www.atomics-sync.com"
    Dim Cap As String = "Help"

    MessageBox.Show(Msg, Cap, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk, MessageBoxDefaultButton.Button1)

End Sub

Private Sub MenuItem12_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MenuItem12.Click
    Application.Exit()
End Sub

End Class
```

Source Code: AtomicSync.vb

```
' Registry class
' -----
' Read values from the registry.
' Modified from MSDN Sample Library Online
' (http://msdn.microsoft.com)

Imports System.Runtime.InteropServices
Imports System.Text

Public Class Registry
    'The base registry keys
    Public Enum RootKey
        ClassesRoot = &H80000000
        CurrentUser = &H80000001
        LocalMachine = &H80000002
        Users = &H80000003
    End Enum

    Class WinApi
#Region " Registry API imports "
        <DllImport("coredll.dll")> _
        Public Shared Function RegOpenKeyEx( _
            ByVal hKey As Int32, ByVal lpSubKey As String, _
            ByVal ulOptions As Int32, ByVal samDesired As Int32, _
            ByRef phkResult As Int32) As Int32
        End Function

        <DllImport("coredll.dll")> _
        Public Shared Function RegCloseKey(ByVal hKey As Int32) As
Int32
        End Function

        <DllImport("coredll.dll")> _
        Public Shared Function RegQueryValueEx( _
            ByVal hKey As Int32, ByVal lpValueName As String, ByVal
lpReserved As Int32, _
            ByRef lpType As Int32, ByVal lpData As Byte(), ByRef lpcbData
As Int32) As Int32
        End Function
#End Region
    End Class

    Private Sub New()
    End Sub

    'Get str val from registry and convert it to string
    Public Shared Function GetString(ByVal rootKey As RootKey, ByVal
keyName As String, ByVal valueName As String) As String
        Dim rslt As String = ""

        Try
            Dim data As Byte() = GetValue(rootKey, keyName, valueName)
            If Not (data Is Nothing) Then
```

```

        rslt = UnicodeEncoding.Unicode.GetString(data, 0,
data.Length)
    End If
    Catch
        'Return empty string
    End Try

    Return rslt
End Function

'Get integer value from registry
Public Shared Function GetInt(ByVal rootKey As RootKey, ByVal
keyName As String, ByVal valueName As String) As Integer
    ' result that is returned
    Dim rslt As Integer = 0

    Try
        ' read specified registry location and convert to int
        Dim data As Byte() = GetValue(rootKey, keyName, valueName)
        If Not (data Is Nothing) Then
            rslt = System.BitConverter.ToInt32(data, 0)
        End If
    Catch
        'Return 0 if error
    End Try

    Return rslt
End Function

Private Shared Function GetValue(ByVal rootKey As RootKey, ByVal
keyName As String, ByVal valueName As String) As Byte()
    ' Returned data
    Dim data As Byte() = Nothing

    ' Registry key handle
    Dim hKey As Int32 = 0

    Try
        'Open registry key
        If WinApi.RegOpenKeyEx(CType(rootKey, Integer), keyName, 0,
0, hKey) = 0 Then
            Dim dataType As Integer
            Dim dataSize As Integer = 0
            WinApi.RegQueryValueEx(hKey, valueName, 0, dataType,
Nothing, dataSize)

            'Read value
            If dataSize <> 0 Then
                data = New Byte(dataSize - 1) {}
                WinApi.RegQueryValueEx(hKey, valueName, 0,
dataType, data, dataSize)
            End If
        End If
    Finally
        If hKey <> 0 Then

```

```
        WinApi.RegCloseKey(hKey)
    End If
End Try

Return data
End Function

End Class
```