A motivation for text on RTP

ALESSANDRO SACCHI

KTH Information and
Communication Technology

Master of Science Thesis
Stockholm, Sweden 2005

IMIT/LCN 2005-18

A motivation for text on RTP

Alessandro Sacchi

28th June 2005

Master of Science Thesis performed at
Wireless@KTH
Stockholm, Sweden

Advisor and Examiner: Gerald Q. Maguire Jr.

Department of Microelectronics and Information Technology (IMIT)
Royal Institute of Technology (KTH)
Stockholm, Sweden

Abstract

The focus of this thesis is on transferring data between mobile devices. It analyzes the resource consumption of using the Wireless LAN interface in some situations, and proposes and evaluates some new applications involving the transfer of voice and data files.

Measurements show the high consumption of battery power due to the operation of a wireless network interface, even if it is not "actively" used to transfer data, while the memory consumption of the running applications is very limited.

This thesis includes also an application to transfer files between two or more Pocket PCs: it is proposed as an addition to the VoIP application Minisip.

Finally I have developed an application in order to explore the possibility to make a voice call by transferring real-time encoded text using UDP and/or RTP streams. This could be used together with speech-to-text and text-to-speech conversion applications at the end points to allow a "voice conversation" even on wireless links with very limited capacity, whereas a standard VoIP conversation would not be affordable.

Sammanfattning

Fokus i detta examensarbete ligger på överföring av data mellan mobila enheter. Det analyserar resursutnyttjandet vid användande av Wireless LAN-anslutning i några situationer, samt brukar analysen för att föreslå och utvärdera ett antal nya applikationer för överföring av röstsamtal och filer.

Mätningar visar den höga konsumtion av batterikraft som anslutning till ett trådlöst nätverk gränsnitt, även då det inte används "aktivt" för dataöverföring, medan de applikationer som använts här behöver bara mycket begränsad minneskapacitet.

I detta examensarbete ingår även ett program för överföring av filer mellan två eller flera mobila apparater, föreslaget att ingå i Minisip, en VoIP applikation.

Dessutom föreslås och utvärderas en applikation som undersöker möjligheterna att vid röstsamtal överföra realtidskodad text via RTP och/eller UDP stream. Använd tillsammans med en röst-till-text- och text-till-röstomvandling i ändpunkterna möjliggör denna genomförande av röstsamtal även då mycket begränsade trådlösa anslutningar nyttjas.

Acknowledgments

I would like to express my sincere gratitude to my project advisor and examiner, Professor Gerald Q. “Chip” Maguire Jr., for the opportunity he gave me to do this master thesis project, and for his continuous and inestimable suggestions, comments, guidance, and also friendship.

I am also very grateful to Professor Stefano Giordano, my supervisor at the University of Pisa, for his teaching and for his help to made possible the development of this joint project.

Special thanks go to all my friends, the ones here in Sweden, and the ones in Italy. Thanks for all the good time we spent together.

And last, I am eternally grateful to my family, who gave me the opportunity of coming here in Stockholm. Thanks for what you have done for me.

Table of contents

List of Figures and Tables

Acronyms

AAL5	ATM Adaptation Layer 5
ASCII	American Standard Code for Information Interchange
CLNP	ConnectionLess Network Protocol
DSSS	Direct Sequence Spread Spectrum
FHSS	Frequency Hopping Spread Spectrum
IANA	Internet Assigned Numbers Authority
IEEE	Institute of Electrical & Electronics Engineers
IETF	Internet Engineering Task Force
IMS	Industrial Scientific Medical (radio spectrum)
IP	Internet Protocol
MAC	Media Access Control
MIKEY	Multimedia Internet KEYing
OFDM	Orthogonal Frequency Division Multiplexing
PCM	Pulse Code Modulation
PSTN	Public Switched Telephone Network
RFC	Request For Comment
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SIP	Session Initiation Protocol
SRTP	Secure Real-time Transport Protocol
TCP	Transport Control Protocol
TDM	Time Division Multiplexing
UDP	User Datagram Protocol
UTF-8	Unicode Transformation Format - 8
WLAN	Wireless Local Area Network

1 Introduction

Some thirty years ago the Internet did not exist. During the mid-1970s digital communications using Pulse Code Modulation (PCM) and Time Division Multiplexing (TDM) became the dominant technologies in enterprise communications systems, and interactive communications were primarily telephone calls over the Public Switched Telephone Network (PSTN).

During the last thirty years, the Internet has grown from a concept to a network that covers the globe. Today nearly everyone in developed countries uses a PC and Internet for both work and entertainment, to communicate with others, and to exchange all types of data. In this environment more and more people are using the Internet to make voice calls via applications that use the PC as an Internet Phone, as an alternative to making voice calls over the PSTN network, along with many features and services that are unavailable via the PSTN.

Mobile devices are playing a more and more important role. Users want small portable devices, with Internet, e-mail and text chatting applications, applications for reading and reviewing documents, a personal organizer, and the ability to make calls to other people, all in the same device. Thus the market offers a lot of devices that combines some or all these features, and the number of new applications for these devices is quickly growing, as they try to satisfy more and more users' requests.

If we think about the possibility to use these portable devices to make voice calls, the most interesting environment for applications regards Voice-over-IP and Voice-over-WLAN technologies. There are two major reasons to use VoIP: lower cost and increased functionality. The attractions of voice as yet another application running over IP are immense in terms of potential cost savings and cost reduction, network convergence, and service innovation.

The main problem in using the Internet for real-time applications, such as a phone call, is the basic architecture of the Internet. Internet is a connectionless packet switched network, packets of data are sent from one side to another, without a reserved physical connection between the sender and the receiver. Of course there are mechanisms to manage the packets during this transfer, but we have no guarantee of their delivery. IP was designed as a best-effort, internetworking protocol, and the Internet does the best it can to deliver packets, but without any assurance.
Moreover, if we think about applications developed for mobile devices, we have to also consider additional problems specifically the consumption of the battery power or the performance of a Wi-Fi connection.

In this thesis I present an application to monitor the status of the mobile device, in particular the battery power and the memory consumption, as these resources can be critical when developing applications for such mobile devices.

It is also proposed and examined a file transfer application, which could be added to the Minisip application to allow users to exchange data files. This application is motivated by the fact that the Pocket PC versions of the widely used Skype and MSN Messenger applications do no support file transfer; while the other versions of these applications do.

The last application presented in this thesis, together with another thesis project (see Johan Sverin's, "Speech Interface for a Mobile Audio Application" [59]), explores the possibility of making voice calls utilizing speech-to-text-to-speech conversion, using RTP protocol for transmitting the real-time "text" representing conversation, which conveniently coincides with the publication of RFC 4103 [56], in order to greatly reduce bandwidth consumption and to ease the design of new real-time audio applications.

2 Background

2.1 User Datagram Protocol

User Datagram Protocol (UDP) [1], [2] is a connectionless transport protocol, used to send messages between two (or more) hosts. It is preferred to the widely used Transport Control Protocol (see section 2.3) when reliability is not the primary target. UDP by itself has no techniques to check if the messages arrive correctly at the destination, as it does not use acknowledgments, nor does it have a field for a sequence number, and it gives no feedback on congestion in the network. Therefore UDP messages can be lost, or they can arrive in the wrong sequence, or faster than the receiver can manage them.

UDP is widely used by real-time applications, when it makes no sense to wait for an acknowledgment, and the upper-layer application can survive the lost of a certain percentage of packets (this percentage depend on the quality the user wants from the application and the nature and distribution of losses); moreover these applications often have techniques to handle packet-loss or out-of-order packets.

Every UDP message is composed by two parts, the ***header*** and the ***data*** field:

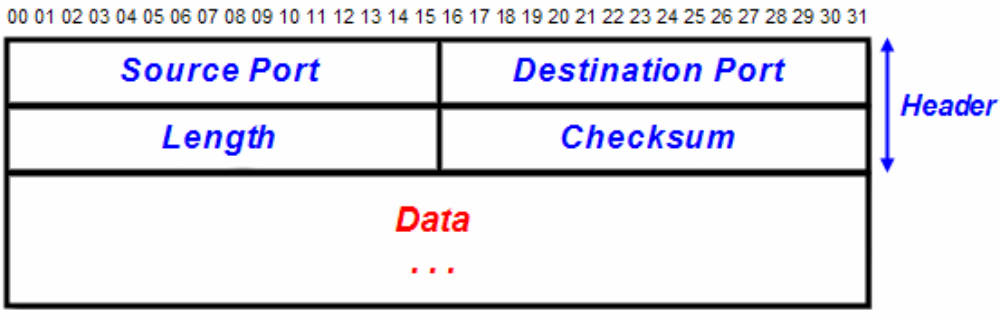

Figure 1: UDP message

The UDP header is divided in four fields, of 16 bits each, that specify the source port number, the destination port number on the remote host, the length (in bytes) of the UDP message, and the optionally checksum, to verify the integrity of the transported data. The UDP data field is an array of bytes of variable length, up to 65536 bytes.

2.2 RTP/RTCP

The Real-time Transport Protocol and the RTP Control Protocol [3], [4], [5] are application layer protocols that IETF [16] has developed to provide additional features to higher layers when they have to support real-time applications. In the majority of the cases they use the User Datagram Protocol (UDP) [1], [2] but RTP/RTCP can be used also with other protocols, such as CLNP [6], AAL5/ATM [7], and others. Figure 2, below, shows the RTP header.

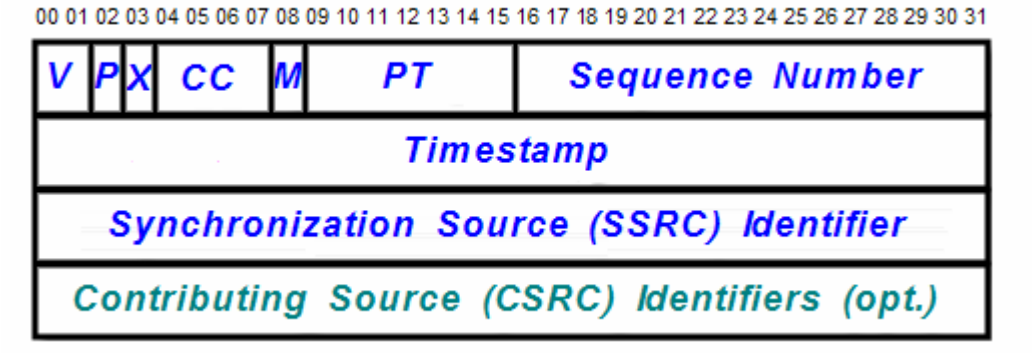

Figure 2: RTP header

Version (V)	**2 bits**	RTP version number
Padding (P)	**1 bit**	If set, the packet contains one or more additional bytes at the end which are not part of the payload
Extension (X)	**1 bit**	If set, indicates the presence of one header extension

CSRC Count (CC)	**4 bits**	Counts the number of identificatori that follow the fixed header
Marker (M)	**1 bit**	Used to mark the frame boundaries in the packet stream
Payload Type (PT)	**7 bits**	Identifies the format of the payload
Sequence Number	**16 bits**	This field is incremented by one for each RTP packet sent. It is used to detect packet loss and to restore the right packet sequence
Timestamp	**32 bits**	Contains the sampling time of the first byte in the RTP data packet. It is used to restore the right temporization synchronizing audio streams and to estimate the jitter
Synchronization Source (SSRC)	**32 bits**	A number to identify the source of a RTP stream
Contributing Source (CSRC) (optional)	**32 bits**	An array of 0 to 15 CSRC elements identifying the contributing sources for the payload

Table 1: RTP header fields

The RTCP packet types are shown in the following Table 2, below.

Sender (SR) and Receiver Report (RR)	Generated by the active sender or receiver conference members (respectively). They have a timestamp filed to estimate the delay and jitter. The first RTCP in a compound packet must always be a report packet
Source Description (SDES)	Gives information on the source
Bye	Indicates that one or more sources are no longer active
Application-Defined (APP)	For experimental use

Table 2: RTCP packet types

2.3 Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) [8], [9], [10] is used by applications that require guaranteed delivery. An application establishes a TCP connection between two endpoints (each endpoint is defined by an IP address and a port number) to transfer data and commands, guaranteeing a reliable transmission using acknowledgments, and timeouts with retransmission. In this technique the destination endpoint sends an acknowledgment message to the sender every time it receives a packet. The sender maintains a copy of the packet sent until it receives the corresponding acknowledgment, then it removes the copy from the transmission buffer. Such as simple implementation (called **Stop and Wait**) has low performance, because it spends a lot of time waiting for the acknowledgment message.

To avoid this problem, TCP introduced the **Sliding Window** technique. It allows transmission of all the packets in the window *without* waiting for the acknowledgments: as the acknowledgments are received, TCP moves the window forward and transmits new packets. Using this implementation it is very important to properly set the size of the window, depending on the round trip time over the link, trying to avoid situations in which the sender sends all the packets in the window, then it waits for a long time for the relative acknowledgments, and then again sends all the new packets in the window, generating an impulsive load in the network.

A lot of other techniques were developed to increase the performance of the TCP, but explaining how they work is not the topic of this thesis.

However, it is interesting to note that TCP protocol also implements a mechanism for congestion control, trying to avoid a situation in which the network traffic exceeds the link capacity and the switching points start on queuing datagrams, increasing the

delay, and in the worst case dropping packets because their buffers are completely full. There are two ways to address the problem of congestion collapse, trying to recover the standard behaviour after a congestion collapse (***Congestion Recovery***), or trying to avoid this situation (***Congestion Avoidance***). For more detail on this topic, see Van Jacobson and Michael J. Karels [11].
The important aspect to note is that the TCP tries to be fair to other applications; this sense of fairness will be used for comparison with UDP based applications, for which the application must provide this fairness, since the UDP transport protocol does not.

2.4 Voice over IP

Voice over IP is a new technology to communicate via the Internet instead of the Public Switched Telephone Network (PSTN). The PSTN works well today, so why should we use a VoIP system? In general phone service via VoIP costs (much) less than the equivalent service via the PSTN. Further you can reduce ongoing expenses in operations and administration of both voice and data communications by combining them within the same organization and network. For users having a computer with an Internet connection and flat rate or a traffic based rate, using VoIP may not involve significant *extra* charges, thus users are able to talk from one corner of the world to another at **very** low cost.
Additionally, VoIP technology enables a lot of new services, for example it is possible to send text messages, images, and video, or to exchange other data (files, shared whiteboards …) between one or more users while they are talking, and so on.

2.4.1 Skype

Skype [12] is an application available for Windows, Mac OS X, Linux, and Pocket PC, to make calls over Internet. It is possible to talk, send instant messages or files

(except for the Pocket PC version), with good sound quality and end-to-end encryption. It is also possible to make calls to a standard phone, although this function (SkypeOut) is not free, while the cost of communication between two PC based Skype users is only the cost of their Internet connection. Although it is not open-source software, it can be downloaded for free. It is very easy and intuitive to use, and more than 125,694,114 users have actually downloaded Skype (as of 26 June 2005).

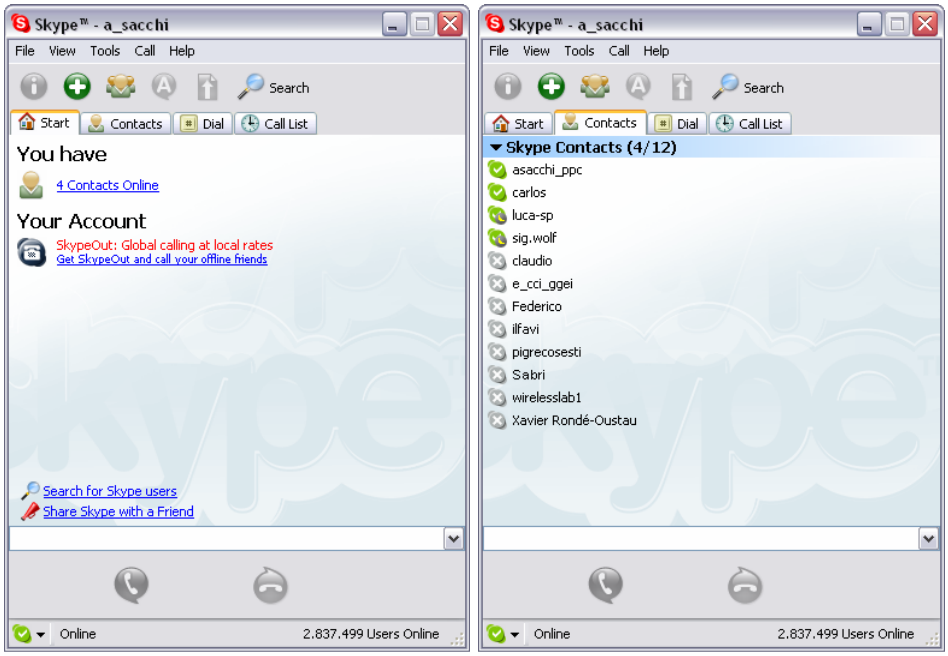

Figure 3: Skype Start window (a) and Contact List window (b) (Caption)

2.4.2 Minisip

Minisip [13] is an application developed at KTH/TSLab running on Linux. It allows users to make calls over Internet, with the additional services of high quality audio and video, multi-party conferencing, and presence information. It is open-source

computer software; thus it is possible to download the source code directly from their web-page [14].

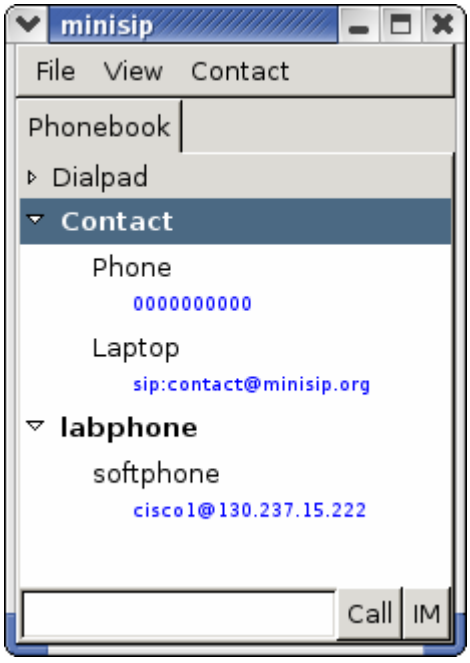

Figure 4: Initial Minisip main window (Caption)

Minisip uses the Session Initiation Protocol (SIP) [15], an IETF (16) standard in VoIP communication, while Skype uses a proprietary protocol; moreover Minisip focuses on open standards and developer communities, while Skype is a commercial product. For more information and a comparison between Skype and Minisip, one can see Carlos Marco Arranz's thesis: IP Telephony, Peer-to-peer versus SIP [17].

2.5 Wireless Local Area Network

A Wireless Local Area Network (WLAN) is a Local area network design to provide wireless connectivity in limited areas, such as buildings and office areas, or a university campus, using high-frequency radio waves or IR. The main WLAN commercial products use the Industrial Scientific Medical (IMS) band, at 2.4 GHz, with spread spectrum techniques [18] to reduce the interferences.

IEEE 802.11 [19] is the standard used in WLAN communication, and it defines the Medium Access Control (MAC) [20] level and the Physical (PHY) level of a local area network with wireless connectivity. In particular 802.11 gives two different ways to implement the Physical level: Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS). Some of the members of this standard are:

- **802.11a :** works in the bandwidth of 5GHz, with transmission speed from 22 Mbps to 54 Mbps at a range of up to 20 meters. It uses Orthogonal Frequency Division Multiplexing (OFDM) [21] modulation.
- **802.11b :** uses the 2.4 GHz band with a speed of 11Mbps, it uses DSSS modulation and the range indoor is around 80-100 meters. The signal modulation uses Differential Phase Shift Keying (DPSK) technique, which combines each bit of the signal with a pseudo-random sequence.
- **802.11g :** uses the 2.4 GHz with transmission speeds up to 54Mbps. It is compatible with 802.11b, but it uses OFDM modulation for higher speeds.
- The **802.11e**, **802.11f**, **802.11h**, **802.11i** standards are under development; 802.11e is focused on Quality-of-Service, and 802.11i on security systems.

2.6 HP iPAQ h5500 Pocket PC

The applications presented in this report run on Microsoft Windows Mobile Pocket PC 2003 OS: for developing and testing I have used a HP iPAQ h5500 Pocket PC (shown in Figure 5).

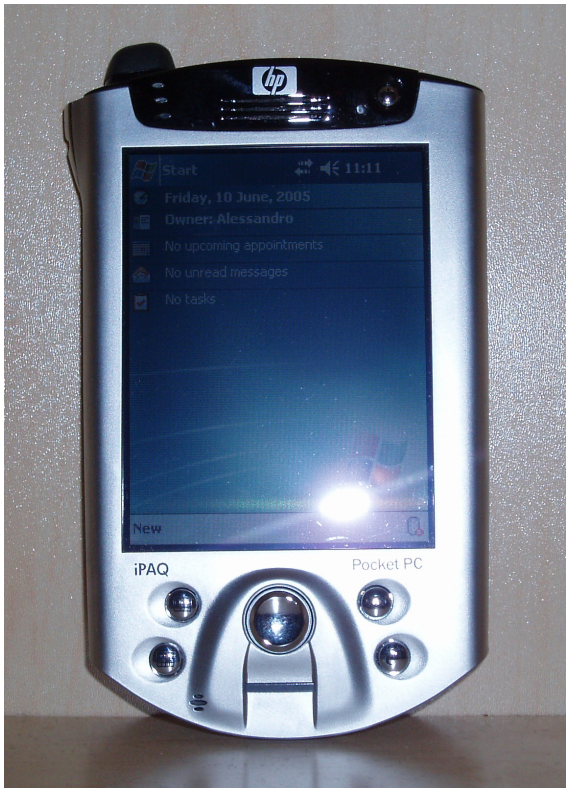

Figure 5: HP iPAQ h5500 Pocket PC

Main specifications of the HP iPAQ h5500 Pocket PC:

Operating System	Microsoft ® Windows ® Mobile™ Pocket PC 2003 Premium
Processor	400 MHz Intel® XScale™ technology-based
Memory	128 MB SDRAM, 48 MB Flash ROM Memory, of which 17 MB are available to the user to store data
Connectivity	Wireless LAN 802.11b, Bluetooth™

Enhanced security	Integrated biometric fingerprint reader: the user can select to require a PIN, password, and/or fingerprint, before log on to the Pocket PC and have access to the data
Battery	Removable/rechargeable 1250 mAh Lithium-ion polymer battery with internal 10-minute bridge battery to maintain data during main battery replacement
Expansion slot	Integrated Secure Digital slot: Secure Digital (SD), Secure Digital Input/Output (SDIO), and MultiMedia Cards (MMC) support
Dimensions (LxWxH)	138 x 84 x 15.9 mm
Weight	206.8 g

Table 3: HP iPAQ h5500 Pocket PC specifications

More information on HP iPAQ h5500 Pocket PC can be found at [22], [23], [24].

The next sections will introduce the software platform that I have used, followed by a description of how to develop an application with a graphical user interface.

2.7 Microsoft .NET

Microsoft .NET [25] is a collection of tools to develop application software for on top of the Windows operating systems. Microsoft's .NET Framework [26] and Visual Studio .NET [27] provide developers with a multi-language platform in which one can create applications for Windows and for Web Services.

Microsoft's .NET Framework [26] has two main components: the Common Language Runtime [28] and the .NET Framework class library [29]. The Common Language Runtime is an agent that manages code at execution time. In particular, it manages memory, thread execution, code execution, code safety verification,

compilation, and other system services. The class library is an object-oriented collection of reusable types that one can use to develop applications, such as command-line applications, or graphical user interface applications, or Web forms and XML Web services.

Microsoft's .NET Compact Framework [30], [31] is a subset of the .NET Framework for portable devices, such as personal digital assistants (PDAs), Pocket PC, and mobile phones. However, some features are missing with respect to the .NET Framework, due to the difference of architecture of the target devices (for more details about this topic see [32], [33].

Visual Studio .NET [27] provides a development environment for creating applications that target the .NET Framework. It supports Visual Basic, C#, J#, and C++ programming languages, and it includes a set of pre-built device profiles which contain information to build application that target specific devices, such as a Windows Application or a Smart Device Application.

2.8 Creating a graphic interface in Microsoft Visual C# .NET

A graphic interface is composed of four fundamental elements:

1. The main window ***form***;
2. The objects in the window, called ***controls***;
3. The ***events*** generated by the controls;
4. The ***methods***, which manage events.

When a control generates an event (i.e. when a user clicks on a button), the related method is automatically executed. To add a control we use the identical syntax of any other declaration, i.e. to declare a control of the *Button* class [34] we could write:

```
private System.Windows.Forms.Button btnStart;
```

After the declaration, we could create an instance of this control using the operator `new`:

```
btnStart = new System.Windows.Forms.Button();
```

Once a control is created, we can set and modify its features by giving the appropriate values to its properties. For example, to define the text label of the button, we could set the value of the **`Text`** property:

```
btnStart.Text = "START";
```

The **position** and the **dimension** of the control are defined in relation to the ***client area*** of the form. The client area is represented by a Cartesian plan whose origin is the upper left hand corner. Inside the client area, every control occupies a position and a rectangular area (control area), whose dimensions are measured in pixels.

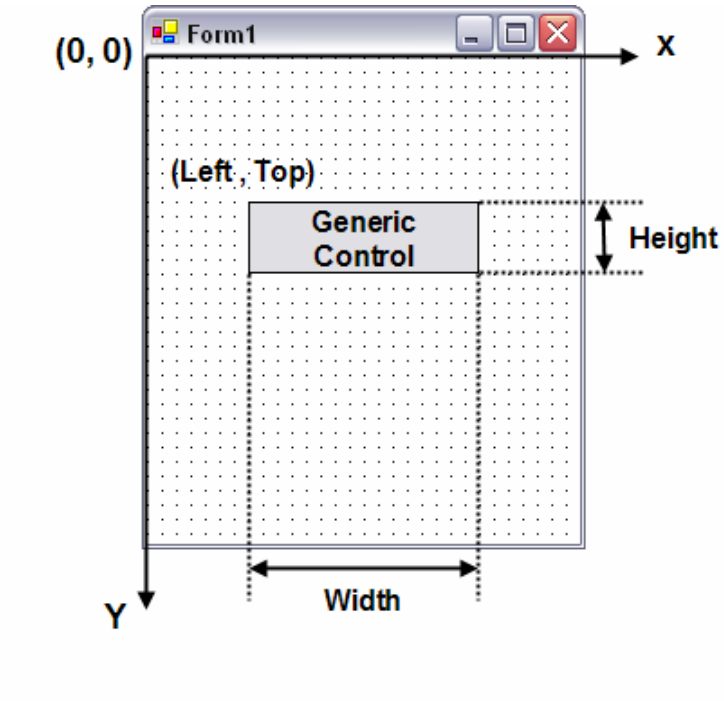

Figure 6: Client Area

The position of the control can be expressed in two ways:

1. Using **Left** and **Top** properties.
2. Using the **Location** property.

Left and **Top** properties store the value of the abscissa (**X**) and the ordinate (**Y**) value, while the **Location** property stores the position of the control using a couple of coordinates, **X** and **Y**, and it can be modified as a single value, whose type is **Point**. The **Location** value reflects the values of **Left** and **Top**.

To define the dimensions of the control area, we could choose one of two ways:

1. Using **Width** and **Height** properties.
2. Using the **Size** property.

Similar to the Location property, the **Size** property stores the dimensions of the control using a couple of values, **Width** and **Height**, and it can be modified as a single object, whose type is **Size**.

For example, we could write these lines to define the position and the size of the button:

```
btnStart.Location = new System.Drawing.Point(48, 64);
btnStart.Size = new System.Drawing.Size(144, 32);
```

Creating a control does not make it operational, nor visible on the screen: it is necessary to add it to the graphic interface, which is a control that belongs to the **Form** type. To do this, we can invoke the **Add()** method:

```
Controls.Add(btnStart);
```

After this command, the button belongs to the graphic interface, and it can now interact with the user. If we want the button to invoke an action when the user clicks on it, we have to write the method we want to be executed, and associate this method with the **`Click`** event. For example, we can define the method to be executed when the user clicks on the button start:

```
private void btnStart_Click(object sender, System.EventArgs e)
{
// Write here the code to be executed when
// a user clicks on the button START
}
```

Then associate this method **`btnStart_Click`** to the event **`Click`** by saying:

```
btnStart.Click += new System.EventHandler(btnStart_Click);
```

A complete example of the creation of a button, and the execution of a method when the user clicks on it, is shown below. In this case, when the user clicks on the button, the application shows a *Message Box* [35] to inform the user that the button Start has been clicked:

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;

namespace EXAMPLE_BUTTON
{
      public class Form1 : System.Windows.Forms.Form
      {
            private System.Windows.Forms.Button btnStart;
```

```
public Form1()
{
      // Required for Windows Form Designer support
      InitializeComponent();
}
// Clean up any resources being used.
protected override void Dispose( bool disposing )
{
      base.Dispose( disposing );
}
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
      this.btnStart = new System.Windows.Forms.Button();
      //
      // btnStart
      //
      this.btnStart.Font = new
            System.Drawing.Font("Microsoft Sans Serif",
            9.75F, System.Drawing.FontStyle.Bold);
      this.btnStart.Location = new
            System.Drawing.Point(48, 64);
      this.btnStart.Size = new System.Drawing.Size(144,
            32);
      this.btnStart.Text = "START";
      this.btnStart.Click += new
            System.EventHandler(this.btnStart_Click);
      //
      // Form1
      //
      this.Controls.Add(this.btnStart);
```

```
            }
            #endregion
            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            static void Main()
            {
                  Application.Run(new Form1());
            }

            private void btnStart_Click(object sender,
                  System.EventArgs e)
            {
                  MessageBox.Show("Button START has been clicked!");
            }
      }
}
```

Figure 7: (a) Graphical interface and (b) the result of clicking the START button

3 Design

First we will examine a simple application which collects local status information (battery usage, memory usage...) to monitor the status of a device. Next we will propose and evaluate an application to transfer data files between two or more devices, using the TCP protocol. Finally, we will propose an application to exchange text via UDP datagrams, to explore the possibility for text over RTP. The text is encoded in UTF-8 standard [54], [55], where UTF-8 stands for Unicode Transformation Format-8. It is an octet (8-bit) lossless encoding of Unicode characters.

The port numbers used in the following applications have been chosen for developing and testing from a slot of unassigned port numbers. The Internet Assigned Numbers Authority (IANA) [60] will assign the port number after the application has been approved.

3.1 Status Client

This application for Pocket PC was developed to collect information on the status of the device, in particular the remaining battery life (in percent), if the device is connected to the AC line or not, the available memory, and the free space in the storage card; in order to monitor the device, user, and application behavior in different situations. For example, one can monitor the power and memory consumption when using Skype [12] (see Section 4.1) or *TCP File Sender* (see Section 3.3 and 4.2), or how much battery power is used when the WLAN interface is always on (see Section 4.3). When a user launches the *Status Client* application (s)he is presented with an interface on the screen as shown in Figure 8.

Figure 8: *Status Client* initial user interface

The user can choose three different options:

1. Storing data in a file: creating a file "YYYYMMDD_log.txt" in the memory in which to store the data. Creating a .txt file enables it to be opened directly from the Pocket PC.
2. Send the data to a remote machine (via UDP messages): in this case the user has to specify the IP address and the port number.
3. Both of them, i.e. both write to the file and send packets via the network.

Finally the user ***has to*** choose the time between status checks; then, the ***Start*** button will be enabled. An example is shown in the figure 9, below, where the user has chosen both options, the packets are to be sent to 130.237.15.224, and status values are collected every 5 seconds.

Figure 9: Starting *Status Client* application (note the use of the popup keyboard to enter values to the program)

3.1.2 Overview of the code (for the complete code see Appendix A.4)

When the user taps on the ***Start*** *Button* [34], the *Status Client* application creates a *timestamp* [36] from the current time, and prepares the ***filePath*** *String* [37] to be the name of the new file.

```
// Get info on the current date
DateTime today = DateTime.Now;
string year = today.Year.ToString();
string month = today.Month.ToString();
string day = today.Day.ToString();
string filePath = (year + month + day + "_log.txt");
```

If the ***store data in a file*** *Check Box* [38] is checked, the *Status Client* creates a new file in the main directory (*My Device*) in the Pocket PC, and writes the first two lines, to describe the data that are going to be written. The "#" char is put at the beginning of the first two lines in order to provide information to Gnuplot [39]. This application, used to plot data, treats these as comments lines, but they serve to label the columns of collected data.

```
if (chkStore.Checked)
{
      StreamWriter sw = File.AppendText(filePath);
      string firstLine = ("#Information about Current Time, Battery
            Power %, AC Line Status, Available Memory (MB) and
            Storage Card Free Space (MB)");
      string secondLine = ("#Time" + "\t\t" + "BPW%" + "\t" + "AC" +
            "\t" + "AvM" + "\t" + "SCFS");
      sw.WriteLine(firstLine);
      sw.WriteLine(secondLine);
      sw.Close();
}
```

The resulting text file looks like:

```
#Information about Current Time, Battery Power %, AC Line Status,
      Available Memory (MB) and Storage Card Free Space (MB)
#Time       BPW%  AC    AvM   SCFS
10:38:33    99    0     54    16
...
```

Then the application begins a loop, in which it checks the current time, the battery status (such as the *Battery Life Percent* value rather than the *Battery Life Time*), if the device is connected to the AC line, the available memory (in Megabytes), and the free

space in storage card (in Megabytes) if one exists (otherwise the free space in the storage memory of the device), and it stores all this information in the *String* ***data*** [37]. To obtain information about the status of the storage card I used the OpenNETCF.IO Namespace [42], which provides the needed functions (see the code in Appendix A.4 between lines 17-203 for more details). This Namespace is part of the open-source Smart Device Framework, which extends the .NET Compact Framework (see also Section 2.7).

```
while ((GetSystemPowerStatusEx(spwrs, false) == 1) && (loop ==
      true))
{
      // Get a timestamp
      DateTime tstamp = DateTime.Now;
      string time = tstamp.TimeOfDay.ToString();

      // Get battery status
      int blp = spwrs.BatteryLifePercent;
      int acl = spwrs.ACLineStatus;
      string battery = blp.ToString();
      string acline = acl.ToString();

      // Get memory status
      MEMORY_STATUS_EX mem = new MEMORY_STATUS_EX();
      double availphy;
      GlobalMemoryStatus(mem);
      availphy = (mem.ullAvailPhys / 1048576); // Convert to Mbytes
      availphy = Math.Round(availphy, 2);
      string availmem = availphy.ToString();

      // Get storage card status
      string[] scardList;
      StorageCard.DiskFreeSpace dfs = new
            OpenNETCF.IO.StorageCard.DiskFreeSpace();
```

```
        scardList = StorageCard.GetStorageCardNames();
        dfs = StorageCard.GetDiskFreeSpace(scardList[0]);
        double freeSpace = (dfs.FreeBytesAvailable / 1048576); //
              Convert to Megabytes
        freeSpace = Math.Round(freeSpace, 2);
        string fs = freeSpace.ToString();
        // Write all the collected info in the string data
        string data = (time + "\t" + battery + "\t" + acline + "\t" +
              availmem + "\t" + fs);
```

If the ***store data...*** checkbox [38] is checked then the string ***data*** is stored in the file created above, and if the ***send data...*** checkbox is checked then the same string is sent to the remote machine in a UDP message. After this the application waits for the number of seconds the user has chosen, and it repeats the loop.

```
        // Store data if related checkbox is checked
        if (chkStore.Checked)
        {
              StreamWriter sr = File.AppendText(filePath);
              sr.WriteLine(data);
              sr.Close();
        }
        // Send data if related checkbox is checked
        if (chkSend.Checked)
        {
              sendBytes = Encoding.UTF8.GetBytes(data);
              client.Send(sendBytes, sendBytes.Length, ipep);
        }
        // Sleep for "sleep" seconds
        Thread.Sleep(sleep);
}
```

Note that using the above functions gives the user the possibility to easily access the file; i.e. the ***File.AppendText*** *Method* [43] creates a ***StreamWriter*** [44] that appends UTF-8 encoded text to an existing file, so the application closes the file after writing the new data, hence the file could be opened by the user at any time.

To stop the application one can to go to the Start → Settings → System → Memory → Running Programs, select **Status Client** and tap ***STOP***, as shown in Figure 10:

Figure 10: Stopping *Status Client* application

3.2 Status Receiver

This application runs on a remote laptop running the Windows OS to receive the information about the mobile device status sent by the *Status Client* application.

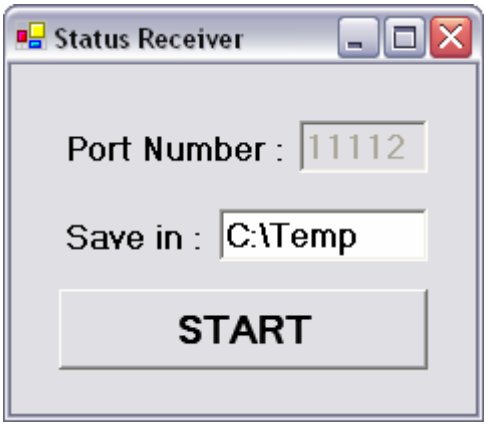

Figure 11: Starting *Status Receiver* application (Caption)

When the user launches the *Status Receiver* application (s)he has to choose the directory (s)he wants to create the file to store incoming data (the directory C:\Temp is suggested by default), and then click on ***START***. Note that it is shown the listening port number (11112 in this example, the same port number used by *Status Client* application), but it is not allowed to change this value from the interface: it can be modified from the source code (see Appendix A.5 for more details).
The application will create a file named *README.txt* in the selected directory to explain the data that will be collected, and it looks like:

```
'Status Receiver' application was started on <YYYY-MM-DD>
The files 'Source_<IPaddress>_onPort_<portNumber>.log' contain
information on the Current Time, Battery Power %, AC Line Status (1
if connected, otherwise 0), Available Memory (in Megabytes) and
Storage Card Free Space (in Megabytes) respectively, of the device
running the 'Status Client' application. i.e.
```

```
Time        BPW%  AC    AvM   SCFS
hh:mm:ss    99    0     53    16
...
```

Then the application starts listening on the port 11112 for incoming messages. When it receives a message sent by the *Status Client* application, it stores the message containing the values of the status of the device in a file named "*Source_<host_IP_Address>_onPort_<Port_Number>.log*".

It is possible to receive data from multiple Pocket PCs at the same time, and the *Status Receiver* application stores the received data in the file corresponding to the source's IP address. The file created is accessible after creation.

```
while(loop)
{
      // listen for client activity on all network interfaces
      // on the selected port number
      IPEndPoint remIpep = new IPEndPoint(IPAddress.Any, port);

      // Receive data
      recvBytes = client.Receive(ref remIpep);

      // Get IPAddress of the remote host
      string hostIP = remIpep.Address.ToString();

      // Write the data in the file
      string filePath = (folder + "\\" + "Source_" + hostIP +
            "_onPort_" + txtPort.Text + ".log");
      sw = File.AppendText(filePath);
      string data = Encoding.UTF8.GetString(recvBytes, 0,
            recvBytes.Length);
      sw.WriteLine(data);
      sw.Close();
}
```

3.3 TCP File Sender

While testing the *Status Client* application on the Pocket PC, I observed there was no possibility to send a file using the widely used MSN Messenger [46] or Skype [12] applications, even though they allow sending files in their PC versions. In particular I could not transfer the locally saved file of status measurements. Therefore, I developed two applications to permit Pocket PC's users to send and receive files using Transmission Control Protocol (TCP) based application. The sender application, and the related receiver (*TCP File Receiver*, see Section 3.4), could be used as they are, but they have no support for security and user authentication. Moreover, it would be interesting to add these features to Minisip [13], where the security problems could be solved by using MIKEY [57] and SRTP [58] protocols, that provide message authentication and key management allowing a secure transfer of the file. Figure 12, below, shows how the *TCP File Sender* application works.

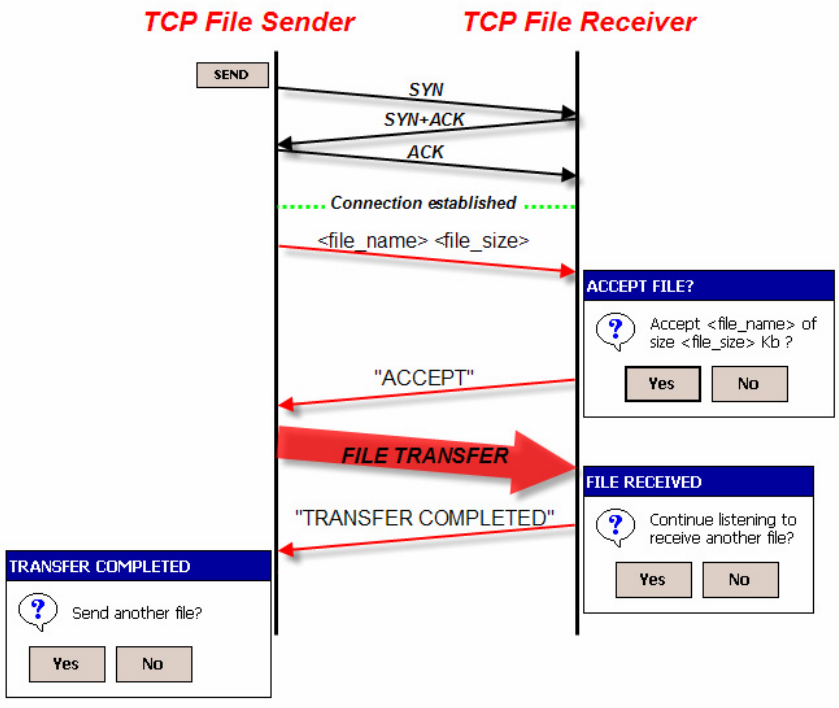

Figure 12: *TCP File Sender* application behavior

When a user starts the *TCP File Sender* application (s)he has to write the IP Address of the machine to which send the file, the name of the file, and tap on the ***Send*** *Button* [34]. If (s)he does not remember the path of the file (s)he wants to send, (s)he can locate it writing a path in the ***Folder*** *Textbox* [45], and then tap on the ***Apply*** *Button*: all the files in the selected folder will be listed in the window below.
An example is shown in Figure 13, below, in which the user wants to send the file ***data.zip*** to 130.237.15.200.

Figure 13: TCP File Sender application example

When the user taps on the ***Send*** *Button*, the *TCP File Sender* application creates a *TCP Client* [48] using the IP Address and the port number of the remote machine (I have chosen to use the port 11115). This constructor automatically attempts a

connection. Then, the application sends a message containing the file name and the size of the file, divided by the null character. This character cannot be used in a file name, so there are no ambiguities to divide efficiently the name from the size in the same *String* [37].

```
// Create a TCP Client and connect
TcpClient client = new TcpClient(txtIPaddr.Text, port);

// ** Send a message with file name and  size **

// Get file name and size
FileInfo fi = new FileInfo(txtFilename.Text);
string fname = fi.Name;
string fsize = fi.Length.ToString();

// Create a string to write these information
// To divide file name from size is used the null char
string finfo = (fname + "\x00" + fsize);

// Get the bytes of the string created above
byte[] finfob = new byte[bsize];
finfob = System.Text.Encoding.ASCII.GetBytes(finfo);

// Get a Network Stream for writing
NetworkStream stream = client.GetStream();

// Send file name and size
stream.Write(finfob, 0, finfob.Length);
```

Then the application waits for the response of the remote machine, which could be “ACCEPT” or “REFUSE”.

```
// Buffer to store the response
byte[] reply = new byte[bsize];
```

```
// String to store the response ASCII representation
string responseData = null;

// Read from the Network Stream to get the response
int bytes = stream.Read(reply, 0, reply.Length);
responseData = System.Text.Encoding.ASCII.GetString(reply, 0,
      bytes);
```

If the application receives an "ACCEPT" message, it starts sending the selected file. If a "REFUSE" message has been received, it prompts with a *Message Box* [35] to inform the local user that the user at the remote machine has refused the file. When the user taps on the ***ok*** *Button* to close the Message Box also the application will be terminated.

Figure 14: *TCP File Sender* "File Refused" Message Box

```
if (responseData == "ACCEPT")
{
      // Send the file
      SendFile(fname, fsize, client, stream);
}

if (responseData == "REFUSE")
{
      // Prompt with a MessageBox
```

```
        MessageBox.Show("File refused!!", "FILE REFUSED",
                MessageBoxButtons.OK, MessageBoxIcon.Exclamation,
                MessageBoxDefaultButton.Button1);

        // Terminate the application
        client.Close();
        Application.Exit();
}
```

In case of "ACCEPTED" message, the application calls the ***SendFile*** method, which opens the selected file, reads the bytes of this file, and writes these bytes in the *Network Stream* [49], to send them to the remote machine.

```
void SendFile(String fname, String fsize, TcpClient client,
NetworkStream stream)
{
        try
        {
                long size = int.Parse(fsize);
                long readBytes = 0;
                int len;

                // Open the file to send
                FileStream fs = new FileStream(txtFilename.Text,
                        FileMode.Open, FileAccess.Read);

                // Loop to read all the bytes of the file
                while(readBytes < size && stream.CanWrite)
                {
                        byte[] datatosend = new byte[bsize];

                        // Read the bytes from the file
                        len = fs.Read(datatosend, 0, datatosend.Length);
```

```
            // Write the bytes on the Network Stream
            stream.Write(datatosend, 0, len);

            // Increase the counter
            readBytes = readBytes + len;
      }
      fs.Close();
```

Then, it waits for a confirmation message sent by the remote machine, to inform the user that the transfer has been completed.

```
byte[] ackb = new byte[bsize];

// String to store the response ASCII representation
string ack = null;

// Read from the Network Stream
int ackBytes = stream.Read(ackb, 0, ackb.Length);
ack = System.Text.Encoding.ASCII.GetString(ackb, 0,
      ackBytes);

// Prompt with MessageBox
DialogResult sendAgain;
sendAgain = MessageBox.Show("Send another file?", ack,
      MessageBoxButtons.YesNo, MessageBoxIcon.Question,
      MessageBoxDefaultButton.Button1);
```

When the *TCP File Sender* application receives this message, it prompts with a *Message Box* to inform the user that the file transfer has been completed, and also to ask for sending another file.

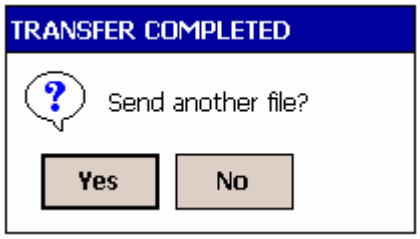

Figure 15: "Transfer completed" Message Box, also asking for sending another file

If the user taps on the ***Yes*** *Button*, the application sends a message to the *TCP File Receiver* application to inform to restart listening for a new file, and then it will be possible to select another file and to send it. On the contrary, if (s)he taps on the ***No*** *Button* the application sends a message to shut down the *TCP File Receiver* application, then it releases all the used resources and exits.

```
if(sendAgain == DialogResult.Yes)
{
      // Send back an "ANOTHER" message
      string an = "ANOTHER";
      byte[] anotherb = new byte[bsize];
      anotherb = System.Text.Encoding.ASCII.GetBytes(an);
      stream.Write(anotherb, 0, anotherb.Length);
}
if(sendAgain == DialogResult.No)
{
      // Send back a "QUIT" message
      string qu = "QUIT";
      byte[] quitb = new byte[bsize];
      quitb = System.Text.Encoding.ASCII.GetBytes(qu);
      stream.Write(quitb, 0, quitb.Length);
      Terminate the application
      client.Close();
      Application.Exit();
}
```

3.4 TCP File Receiver

This application for Pocket PC was developed to receive files sent by the *TCP File Sender* application (see Section 3.3).

Figure 16: *TCP File Receiver* initial user interface

When the user taps on the ***Start*** *Button* the application checks if the selected directory to store incoming files exists (the default directory is *\My Documents\MyFiles*); if not, the application creates this directory. Also, the user can taps on the ***Apply*** *Button*: the ***Polulate*** method is called (see the entire code in Appendix A.7 for more details), and all the files locating in the selected directory will be listed in the window below.

Then the application creates a *TCP Listener* [50] to listen for and accept incoming connection requests from *TCP Clients* [48]. Once a *TCP Listener* object is defined,

we need to invoke its *Start* method [51] to make it begin waiting for incoming requests.

```
// Set up a TCP Listener
IPAddress myIP = IPAddress.Parse("0.0.0.0");
TcpListener server = new TcpListener(myIP, port);

// Start listening for client request
server.Start();

// Perform a blocking call to accept request
TcpClient client = server.AcceptTcpClient();
```

Once the connection with the client is established, the application is ready to receive data. The first data sent by the client is a string containing the name and the size of the file the client wants to send, divided by the null character. The application receives this data, and splits the string in two pieces, using the null character. Then, it converts to Kilobytes the size of the file, and prompts the user with a *Message Box* [35] to inform of both the file name and size (s)he may receive, to allow him/her to accept or refuse the file.

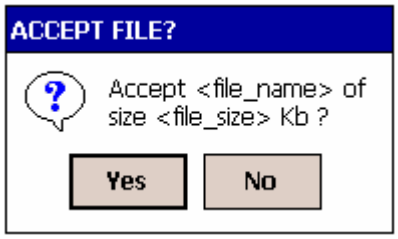

Figure 17: *TCP File Receiver* "Accept File?" Message Box

```
// Get a Network Stream for reading and writing
NetworkStream stream = client.GetStream();
```

```
// Buffer for reading data
byte[] bytes = new byte[bsize];
string data = null;

// Read from the stream
int i = stream.Read(bytes, 0, bytes.Length);
data = System.Text.Encoding.ASCII.GetString(bytes, 0, i);

data = data.Substring(0, i);
char nullchar = nul[0];
string[] finfo = data.Split(nullchar);
string fname = finfo[0].ToString();
string fsize = finfo[1].ToString();

// Convert bytes to Kilobytes to display
double fsizeKB = int.Parse(fsize);
fsizeKB = (fsizeKB / 1024);
fsizeKB = Math.Round(fsizeKB, 0);
string fsKB = fsizeKB.ToString();

// Prompt with a MessageBox to choose if accept or refuse the file
DialogResult accept;
accept = MessageBox.Show("Accept " + fname + " of size " + fsKB + "
      Kb ?", "ACCEPT FILE?", MessageBoxButtons.YesNo,
      MessageBoxIcon.Question, MessageBoxDefaultButton.Button1);
```

If the user wants to accept the file, and taps on the ***Yes*** *Button*, the application sends a reply to the client to inform them that the receiving user has agreed to accept the file, and the client can begin sending data, then it invokes the ***ReceiveFile*** method to receive the incoming bytes. If the user refuses the file, by tapping on the ***No*** *Button*, the application sends a reply to the client to inform them that the file has been refused.

```
if(accept == DialogResult.No)
{
      // Send back a "REFUSE" reply
      string reply = "REFUSE";
      byte[] replyb = new byte[bsize];
      replyb = System.Text.Encoding.ASCII.GetBytes(reply);
      stream.Write(replyb, 0, replyb.Length);

      // Terminate the application
      client.Close();
      Application.Exit();
}

if(accept == DialogResult.Yes)
{
      // Send back an "ACCEPT" reply
      string reply = "ACCEPT";
      byte[] replyb = new byte[bsize];
      replyb = System.Text.Encoding.ASCII.GetBytes(reply);
      stream.Write(replyb, 0, replyb.Length);

      // Start receiving the file
      ReceiveFile(fname, fsize, client, server, stream);
}
```

The ***ReceiveFile*** method creates the file to write the received bytes, and then starts a loop to receive all the bytes sent by the client. When the entire file has been received, it sends a confirmation message to the client, to inform them that the file has been correctly received and the file transfer is terminated.

```
void ReceiveFile(String fname, String fsize, TcpClient client,
      TcpListener server, NetworkStream stream)
{
```

```
try
{
        long size = int.Parse(fsize);
        long readBytes = 0;

        // Create the file to write the received bytes
        FileStream fs = new FileStream(txtLocaldir.Text + "\\" +
                fname, FileMode.Create, FileAccess.Write);

        // loop to receive all the data
        while(readBytes < size)
        {
                byte[] buffer = new byte[bsize];

                // Read from the Network Stream
                int j = stream.Read(buffer, 0, buffer.Length);

                // Write in the File Stream
                fs.Write(buffer, 0, j);

                // Increase the counter
                readBytes = readBytes + j;
        }

        // Close the file
        fs.Close();

        // Send a "TRANSFER COMPLETED" message
        string ok = "TRANSFER COMPLETED";
        byte[] fileok = new byte[bsize];
        fileok = System.Text.Encoding.ASCII.GetBytes(ok);
        stream.Write(fileok, 0, fileok.Length);
```

Then, the application prompts with a *Message Box* [35] to inform the local user the file has been received, and also asking if continue listening for another file. If the

user taps on the ***Yes*** *Button*, the TCP File Receiver application starts listening for a message from the client. In this case, if the incoming message is "ANOTHER", it restarts listening for receiving the name and the size of the file, while if the incoming message is "QUIT" (sent by the client) or any other message else than "ANOTHER", the application releases all the used resources and exits.

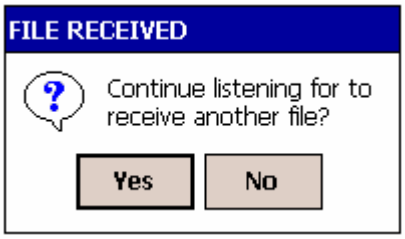

Figure 18: "File Received" TCP File Receiver Message Box

```
DialogResult again;
again = MessageBox.Show("Continue listening to receive another
      file?", "FILE RECEIVED", MessageBoxButtons.YesNo,
      MessageBoxIcon.Question, MessageBoxDefaultButton.Button1);

if(again == DialogResult.Yes)
{
      // Buffer for reading data
      byte[] by = new byte[bsize];
      string str = null;

      // Read from the stream
      int j = stream.Read(by, 0, by.Length);
      str = System.Text.Encoding.ASCII.GetString(by, 0, j);

      if(str == "ANOTHER")
      {
            server.Stop();
            client.Close();
            StartServer();
```

```
        }

        if(str == "QUIT")
        {
                server.Stop();
                client.Close();
                MessageBox.Show("Shutting down application", "QUIT
                        MESSAGE RECEIVED", MessageBoxButtons.OK,
                        MessageBoxIcon.Exclamation,
                        MessageBoxDefaultButton.Button1);
                Application.Exit();
        }

        server.Stop();
        client.Close();
        Application.Exit();
}

if(again == DialogResult.No)
{
        server.Stop();
        client.Close();
        Application.Exit();
}
```

3.5 UDP and RTP Text applications

The following application explores the possibility to communicate from a speech-to-text conversion application on the source side (see [59]), sending text instead of a stream of voice samples, and feeding them to a text-to-speech conversion application on the receiver side. The *UDP Text "Char"* application, presented in the Section 3.5.1, below, is a simple application to send and receive UDP messages containing only one character each. Besides in Section A.9 is presented the source code of an application that implements the standard proposed in the Request For Comment 4103 [56], on carrying text conversation in RTP packets: even if at the time of writing this application is not complete, it represents a base for continual research.

3.5.1 UDP Text "Char" application

I have encountered several problems developing this application, due to the UDP client [40] behavior: although it works, it is not bug-free. In particular, on exiting, it catches an exception on UDP socket, and then exits. This bug is related to the fact the *UDP Client* [40] listens for new messages in a loop, and the invoked *Receive Method* [52] blocks until a datagram arrives from a remote host. When the user wants to exit the application and taps on *Menu File => Exit* [53] the *UDP Client* is still listening for new messages, and this event catches the exception.

When starting the application, the user has to enter the IP address of the machine with which (s)he wants to communicate, and tap on ***Start** Button* [34]. The *UDP Text "Char"* application waits until a character is typed in the *Outgoing Text Box* [45], then encodes it in a UTF-8 format [54], [55], and sends the encoded message to the selected IP address, using the port number 11117.

Figure 19: *UDP Text "Char"* application

```
void Send()
{
      try
      {
            // Encode and send the message
            msg = txtOutgoing.Text;
            bytes = Encoding.UTF8.GetBytes(msg);

            string remIP = txtIPaddr.Text;
            client.Send(bytes, bytes.Length, remIP, port);
            txtOutgoing.Text = "";
      }
      catch(Exception es)
      {
            MessageBox.Show(es.Message, "SEND EXCEPTION");
```

```
        }
}
```

At the same time, using a different thread, the application listens for incoming messages from the selected IP address on the same port number, and when it receives a message it shows the text in the *Incoming Text Box* [45], shifting the letters from right to left on the screen.
It is also possible to store all the incoming messages in a text file, created in the directory *My Device\My Documents*, to save the conversation history.

```
void Receive()
{
     try
     {
            // Set up IPEndPoint
            IPAddress ipAddr = IPAddress.Parse(txtIPaddr.Text);
            IPEndPoint ipep = new IPEndPoint(ipAddr, port);

            // Create the file to store incoming messages
            if(chkStore.Checked)
            {
                  sw = File.AppendText("\\My Documents\\" +
                        txtIPaddr.Text + "-UDP_TEXT_CHAR_log.txt");
                  sw.WriteLine(" ");
            }

            while(loop)
            {
                  // Receive string
                  bytes = client.Receive(ref ipep);
                  msg = Encoding.UTF8.GetString(bytes, 0,
                        bytes.Length);
```

```
                    // Show the string
                    label14.Text = label13.Text;
                    label13.Text = label12.Text;
                    ...
                    label2.Text = label1.Text;
                    label1.Text = msg;

                    // Store the string
                    if(chkStore.Checked)
                    {
                          sw.Write(msg);
                    }
              }
        }
        catch(Exception er)
        {
              MessageBox.Show(er.Message, "RECEIVE EXCEPTION");
        }
}
```

If using this application in combination with a synthesizer, it will be necessary to buffer the incoming text in a string before feeding the text-to-speech engine, because they usually accept as input a word or a set of words, and not single characters.

4 Measurements results

A series of measurements where made. The first of these was the monitoring of the device status while running Skype [12], to obtain the battery power and memory consumption of using this VoIP application to make a voice call. Then, we measured the battery consumption of using the Wireless LAN interface, while sending a big amount of data, or small packets, or having the WLAN interface turned ON but sending nothing. Finally, we measeured the bandwidth consumption while using Skype and UDP Text "Char" application to send a short text.

4.1 Resource consumption using Skype

We have checked the resources consumption every 30 seconds while the user is running Skype [12]. The data is stored both at the Pocket PC and sent to a remote machine. When the Skype application terminates we continue to monitor the status of the device before turning off the WLAN interface and after having turned it off.

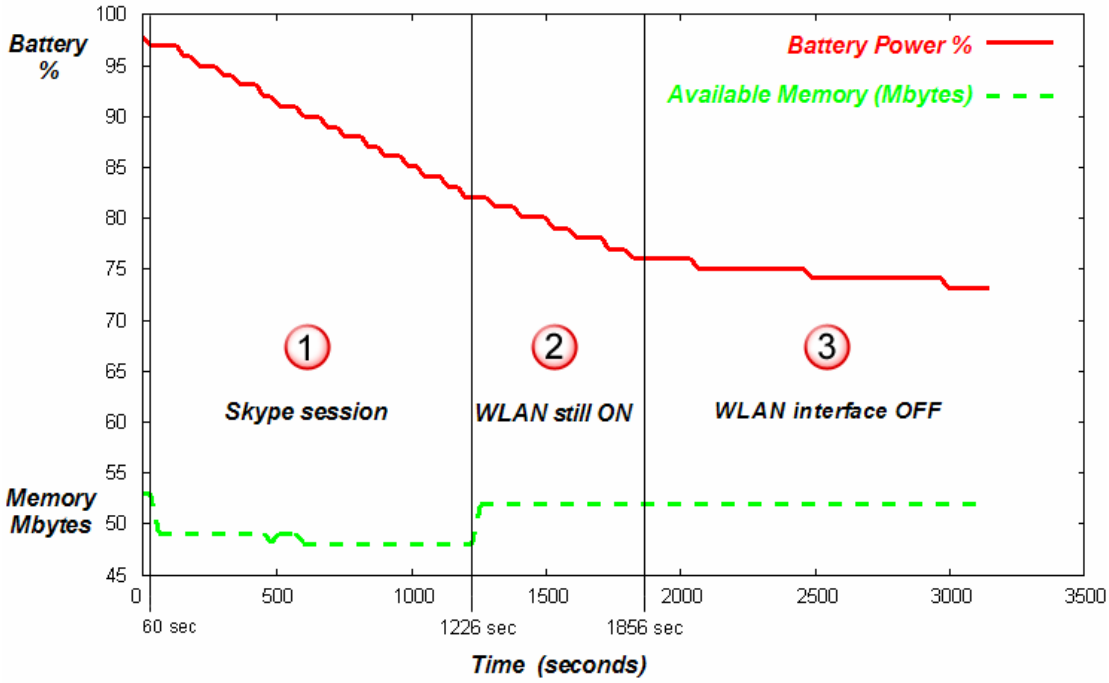

Figure 20: Resources consumption using Skype

→ **at 60 sec:** run Skype and begin a call.

→ **at 1226 sec:** (~ 17 minutes after the beginning of the call): terminate the call and close Skype.

→ **at 1856 sec:** turn off the WLAN interface.

From Figure 20 it is possible to see the state of the Pocket PC in these three different situations: during the Skype session (1), when Skype is closed but the WLAN interface is still turned on (2), and when it is turned off (3). When Skype terminates the slope of the battery consumption line (red) changes just a bit (from one percentage point minus every ~ 77.73 seconds to every ~ 105 seconds), but there is a big improvement in the available memory line (green): as all the memory resources used are released. Turning off the WLAN interface has no influence on the memory consumption, but it has a big influence on the battery consumption, and the battery life decreases much more slowly, from one percentage point minus every ~ 105 seconds to every ~ 431 seconds.

4.2 Battery power consumption using WLAN interface

In this measurement we examine the resource consumption of using *TCP File Sender* and *TCP File Receiver* applications, monitoring the behavior of the Pocket PC while transferring a file of ~ 54 Megabytes each. The graph shown below (Figure 21) shows the influence of the WLAN interface operations on the battery power consumption; we see that it has very little dependence on the amount of data sent. The power consumption is nearly the same if sending a big size file (red), or sending UDP datagrams containing the information on the device status (about 62 bytes each packet, sent every 100 milliseconds) (green), or having the WLAN interface on without sending anything (blue).

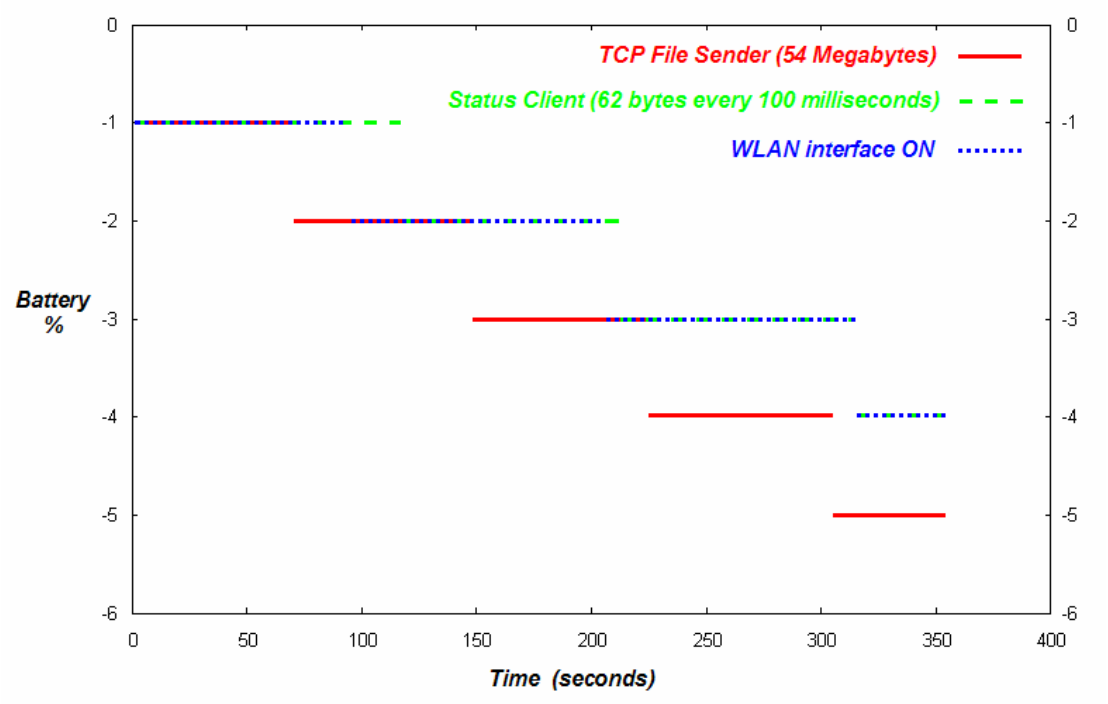

Figure 21: Battery power consumption using WLAN interface with different loads

4.3 Battery power consumption having WLAN interface always ON

In this measurement we examine the battery power consumption when the WLAN interface is always on, repeating the test with measurements every 30, 60, or 120 seconds. The results are very similar and confirm the previous analysis: thus having the WLAN interface always on is extremely onerous for the battery life, even if it does **not** transmit anything.

At (1), after 9783 seconds (nearly 2 hours and 43 minutes) the battery life reaches the 15% value, at this point the Pocket PC's OS automatically turns off the WLAN interface (see the red arrow in Figure 22). After 5 hours the battery power reaches the 0% value.

Having the WLAN interface always on reduces the battery power by one percentage point every nearly 120 seconds, while when the WLAN interface is turned off by the

OS the battery power reduces by one percentage point every ~ 8 minutes and 36 seconds. Hypothetically, if the OS does not turn of the WLAN interface, the battery power would reach the 0% value after 11586 seconds, nearly 3 hours and 13 minutes (see (2) in Figure 22 below), vs. an idle operating time of ~ 4 hours and 52 minutes.

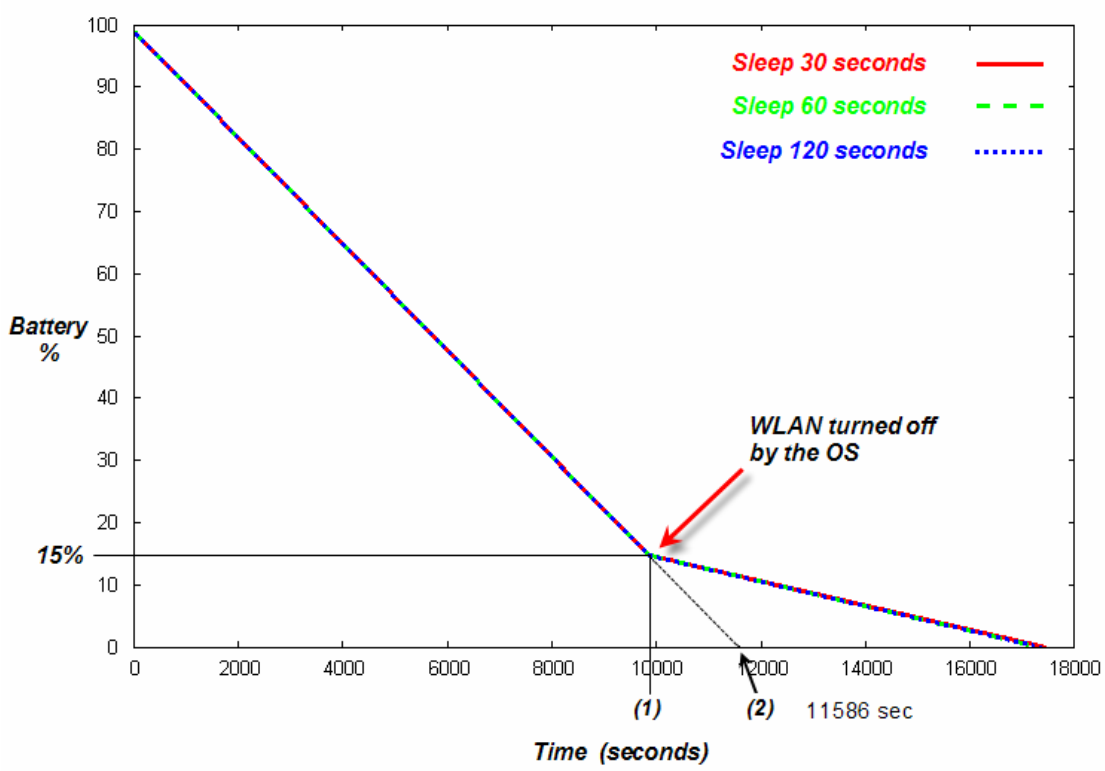

Figure 22: Battery power consumption with WLAN interface always ON
At (1) the WLAN interface is turned off by the OS.

Comparing the previous results with the battery power consumption when the WLAN interface is turned off shows the large amount of battery power used by the WLAN interface (Figure 23, below):

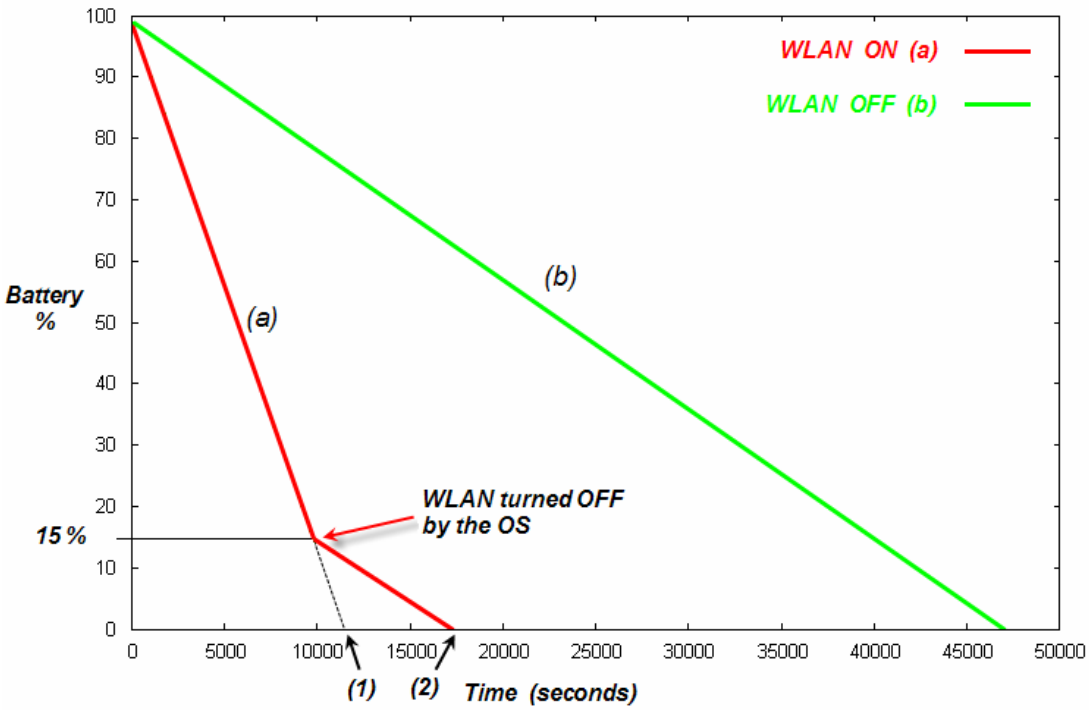

Figure 23: Battery power consumption with WLAN always ON (a) or OFF (b)

When the WLAN interface is **OFF** the battery power will reach the 0% value after nearly 13 hours and 8 minutes, while if the WLAN interface is always **ON** the battery power is exhausted in about 5 hours, 38% of the previous time. In this test I have considered the effective time at which the battery power reaches 0% (shown as (2) in figure 23), because in reality the Pocket PC will automatically turn off the WLAN interface when the battery power falls to 15%. Considering that when the WLAN is on the battery power decreases one percent every 2 minutes, it is possible to extrapolate the time at which the battery power would reach 0% value if the OS would not turn off the WLAN interface, and this time is nearly 3 hours and 13 minutes (1).

It is also interesting that if the WLAN is turned off from the beginning of the test (red line) the battery power decreases one percent every ~ 7 minutes and 53 seconds,

while if the WLAN is turned off by the Pocket PC's OS, in the last 15% of the battery life, the battery power decreases by one percent every ~ 8 minutes and 36 seconds (the green line between 15% and 0% and the red line of Figure 23 are **not** parallel). This because probably the OS also turns off the *Wireless Control* program (and perhaps also other application to consume the less possible the battery), but from the values collected by the Status Client application it has not been possible to determine the cause of this behavior.

4.4 Bandwidth consumption of UDP Text "Char" vs. Skype

In this measurement I have used the UDP Text "Char" application (see Section 3.5.1) to transmit the abstract of this thesis, to check the bandwidth consumption when sending text on UDP datagrams. Sniffing the traffic on the Wireless LAN interface with Ethereal [61], a network protocol analyzer, I obtained a value of 90270 bytes sent.

Then I have used Skype [12] to send the same text, speaking in a microphone and reading the abstract of this thesis, and the resulting traffic value was 608662 bytes.

We can see there is a big difference in bandwidth consumption while using these two applications: sending text on UDP datagrams needs nearly **15%** of the bandwidth necessary to transmit the same information using Skype.

The disadvantage is that we need to use speech-to-text and text-to-speech conversion applications at the endpoints, and the quality of the speech depends much more from the speech synthesis engines. To transmit information we can use basic text-to-speech and speech recognition engines, but adding emotional markup and voice profile information would greatly improve the quality experienced from the users.

5 Conclusions

In this thesis we have introduced some applications with the intentoin to support the developers that work with mobile devices, offering applications to monitor the status of devices and to exchange files and data. In particular, monitoring resource consumption is fundamental when working with devices with limited resources in terms of computational power and memory, comparing to a standard PC. However, the big limitation of portable devices is their battery, and this thesis highlights that the use of the WLAN interface is extremely deleterious for the battery lifetime, even without "actively" using the network interface.

Additionally an application has been developed for transferring file a/from portable devices. The application is ready-to-use for testing and in developing environments, but lacks security and authentication.

Finally, this thesis explores the possibility to communicate using real-time encoded text on a RTP stream [3], [4], [5] instead of sending an encoded audio stream, reducing bandwidth while potentially increasing redundancy allowing better error detection and correction. The disadvantage is greater complexity in the end point devices that must convert speech-to-text and text-to-speech during a real-time communication session, despite the lack of computational power and memory resources of a portable device. The text transfer applications are not completely bug-free, but they represent a simple base for continual research and development. This is particularly interesting considering the publication of the IETF Request For Comment 4103 (June 2005) [56] that proposes a standard for text over RTP communication.

6 Open issues and future work

Adding security and authentication support in the *TCP File Sender* and *Receiver* applications is clearly essential for further work. It would be interesting to add the features provided by there applications to Minisip [13] to solve the security problems by using MIKEY and SRTP, but other methods also could be explored.

Moreover the publication of the RFC 4103 opens a new research area that would be interesting to explore looking for new techniques to improve performances of Voice-over-IP applications in mobile environment. An important open issue regarding the development of applications that satisfy the requirements of the RFC 4103 is to address the issue of TCP "friendliness": i.e. fairness in the sense of a TCP stream for text over RTP.

7 References*

[1] J. Postel. User Datagram Protocol. RFC 768. IETF, 28 August 1980.
http://www.ietf.org/rfc/rfc0768.txt

[2] Network Sorcery RFC Sourcebook. UDP, User Datagram Protocol. Last updated 22 November 2004.
http://www.networksorcery.com/enp/protocol/udp.htm

[3] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550. IETF, July 2003.
http://www.ietf.org/rfc/rfc3550.txt

[4] Network Sorcery RFC Sourcebook. RTP, Real-time Transport Protocol. Last updated 14 May 2005.
http://www.networksorcery.com/enp/protocol/rtp.htm

[5] Network Sorcery RFC Sourcebook. RTCP, RTP Control Protocol. Last updated 27 May 2005.
http://www.networksorcery.com/enp/protocol/rtcp.htm

[6] Network Sorcery RFC Sourcebook. CLNP, Connectionless Network Protocol, ISO 8473. Last updated 21 January 2004.
http://www.networksorcery.com/enp/protocol/clnp.htm

[7] Network Sorcery RFC Sourcebook. AAL5, ATM Adaptation Layer 5. Last updated 18 January 2004.
http://www.networksorcery.com/enp/protocol/aal5.htm

* All the websites were last accessed on June 26, 2005.

[8] Information Sciences Institute. University of Southern California. Transmission Control Protocol. RFC 793. Defense Advanced Research Projects Agency. September 1981.
http://www.ietf.org/rfc/rfc0793.txt

[9] Network Sorcery RFC Sourcebook. TCP, Transmission Control Protocol. Last updated 22 April 2005.
http://www.networksorcery.com/enp/protocol/tcp.htm

[10] John Kristoff. The Transmission Control Protocol. April 2000.
http://condor.depaul.edu/~jkristof/technotes/tcp.html

[11] V. Jacobson and M. Karels. Congestion Avoidance and Control. Proceedings of the SIGCOMM 1988 Symposium. November 1988.
http://www-nrg.ee.lbl.gov/papers/congavoid.pdf

[12] Skype - The whole world can talk for free home page. Last updated 26 June 2005.
http://www.skype.com/

[13] Minisip Introduction page. Last updated 26 June 2005.
http://www.minisip.org/index.html

[14] Minisip Download page. Last updated 26 June 2005.
http://www.minisip.org/download.html

[15] J. Rosenberg, H. Schulzrinne, et al. SIP: Session Initiation Protocol. RFC 3261. IETF, June 2002.
http://www.ietf.org/rfc/rfc3261.txt

[16] The Internet Engineering Task Force home page. Last updated 22 June 2005.
http://www.ietf.org/

[17] Carlos Marco Arranz. IP Telephony: Peer-to-peer versus SIP. June 2005.
ftp://ftp.it.kth.se/Reports/DEGREE-PROJECT-REPORTS/

[18] Jacobus Petrus Franciscus Glas. The principles of Spread Spectrum communication. December 1996.
http://cas.et.tudelft.nl/~glas/ssc/techn/techniques.html

[19] IEEE 802.11 Wireless Local Area Networks - The Working Group for WLAN Standards. Last updated 18 February 2005.
http://grouper.ieee.org/groups/802/11/

[20] A. Chandra, V. Gummalla, and John O. Limb. Georgia Institute of Technology. Wireless Medium Access Control Protocols. Last updated 26 June 2005.
http://www.comsoc.org/livepubs/surveys/public/2q00issue/gummalla.html

[21] Orthogonal Frequency Division Multiplexing (OFDM) forum. Last updated 26 June 2005.

http://www.ofdm-forum.com/

[22] Gerald Q. Maguire Jr.'s notes on using the HP iPAQ h5550. Last updated 26 June 2005.

http://web.it.kth.se/~maguire/ipaq-notes.html

[23] HP iPAQ h5500 Pocket PC series. Last updated 26 June 2005.

http://h20000.www2.hp.com/bizsupport/TechSupport/DocumentIndex.jsp?locale=en_US&contentType=SupportManual&docIndexId=3124&prodTypeId=215348&prodSeriesId=322916&lang=en&cc=us

[24] HP iPAQ Pocket PC h5500 Series - World Wide QuickSpecs. Last updated 26 June 2005.

http://h18002.www1.hp.com/products/quickspecs/11646_div/11646_div.HTML

[25] Microsoft .NET: driving business value with the Microsoft Platform. Last updated 26 June 2005.

http://www.microsoft.com/net/

[26] Microsoft .NET Framework Developer Center. Last updated 26 June 2005.

http://msdn.microsoft.com/netframework/

[27] Microsoft Visual Studio Developer Center. Last updated 26 June 2005.

http://msdn.microsoft.com/vstudio/

[28] The Common Language Runtime (CLR). Last updated 26 June 2005.

http://msdn.microsoft.com/netframework/programming/clr/default.aspx

[29] Microsoft .NET Framework Class Library. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/cpref_start.asp

[30] Microsoft .NET Compact Framework. Last updated 26 June 2005.

http://msdn.microsoft.com/smartclient/understanding/netcf/

[31] Microsoft .NET Compact Framework QuickStart Tutorial. Last updated 26 June 2005.

http://samples.gotdotnet.com/quickstart/compactframework/

[32] Microsoft .NET Compact Framework Technical Articles. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetcomp/html/CompactfxTechArt.asp

[33] Getting Started with Visual Studio .NET and the Microsoft .NET Compact Framework. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetcomp/html/netcfgetstarted.asp

[34] MSDN Library. Microsoft .NET Framework Class Library: Button Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemwebuiwebcontrolsbuttonclasstopic.asp

[35] MSDN Library. Microsoft .NET Framework Class Library: MessageBox Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemwindowsformsmessageboxclasstopic.asp

[36] MSDN Library. Microsoft .NET Framework Class Library: DateTime Structure. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemdatetimeclasstopic.asp

[37] MSDN Library. Microsoft .NET Framework Class Library: String Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemstringclasstopic.asp

[38] MSDN Library. Microsoft .NET Framework Class Library: CheckBox Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemwindowsformscheckboxclasstopic.asp

[39] Gnuplot homepage. Last updated 1 May 2005.

http://www.gnuplot.info/

[40] MSDN Library. Microsoft .NET Framework Class Library: UdpClient Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemnetsocketsudpclientclasstopic.asp

[41] MSDN Library. Microsoft .NET Framework Class Library: IPEndPoint Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemnetipendpointclasstopic.asp

[42] OpenNETCF.org - The Premier .NET Compact Framework Shared Source Site. Last updated 26 June 2005.

http://www.opennetcf.org/CategoryView.aspx?category=Home

[43] MSDN Library. Microsoft .NET Framework Class Library: File.AppendText Method. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemiofileclassappendtexttopic.asp

[44] MSDN Library. Microsoft .NET Framework Class Library: StreamWriter Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemiostreamwriterclasstopic.asp

[45] MSDN Library. Microsoft .NET Framework Class Library: TextBox Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemwindowsformstextboxclasstopic.asp

[46] MSN Messenger home page. Last updated 26 June 2005.

http://messenger.msn.com/

[47] J. Rosenberg, H. Schulzrinne, et al. Session Initiation Protocol (SIP) Extension for Instant Messaging. RFC 3428. IETF, December 2002.
http://www.ietf.org/rfc/rfc3428.txt

[48] MSDN Library. Microsoft .NET Framework Class Library: TcpClient Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfSystemNetSocketsTcpClientClassTopic.asp

[49] MSDN Library. Microsoft .NET Framework Class Library: NetworkStream Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfSystemNetSocketsNetworkStreamClassTopic.asp

[50] MSDN Library. Microsoft .NET Framework Class Library: TcpListener Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemnetsocketstcplistenerclasstopic.asp

[51] MSDN Library. Microsoft .NET Framework Class Library: TcpListener.Start Method. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemnetsocketstcplistenerclassstarttopic.asp

[52] MSDN Library. Microsoft .NET Framework Class Library. UdpClient.Receive Method. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemnetsocketsudpclientclassreceivetopic.asp

[53] MSDN Library. Microsoft .NET Framework Class Library. MainMenu Class. Last updated 26 June 2005.

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfSystemWindowsFormsMainMenuClassTopic.asp

[54] F. Yergeau. UTF-8, a transformation format of ISO 10646. RFC 3629. IETF, November 2003.

http://www.ietf.org/rfc/rfc3629.txt

[55] UTF-8 and Unicode Standards. Last updated 10 February 2005.

http://www.utf-8.com/

[56] G. Hellstrom and P. Jones. RTP Payload for Text Conversation. RFC 4103. IETF, June 2005.

http://www.ietf.org/rfc/rfc4103.txt

[57] J. Arkko, E. Carrara, et al. MIKEY: Multimedia Internet KEYing. RFC 3830. IETF, August 2004.

http://www.ietf.org/rfc/rfc3830.txt

[58] M. Baugher, D. McGrew, et al. The Secure Real-time Transport Protocol (SRTP). RFC 3711. IETF, March 2004.

http://www.ietf.org/rfc/rfc3711.txt

[59] Johan Sverin. Speech Interface for a Mobile Audio Application. June 2005.

ftp://ftp.it.kth.se/Reports/DEGREE-PROJECT-REPORTS/

[60] Port Numbers. Internet Assigned Numbers Authority (IANA). Last updated 23 June 2005.

http://www.iana.org/assignments/port-numbers

[61] Ethereal - The world's most popular network protocol analyzer home page. Last updated 26 June 2005.

http://www.ethereal.com/

A Appendix

A.1 Status of the device during the Skype-call test

#Information about Current Time, Battery Power %, AC Line Status, Available Memory (MB) and Storage Card Free Space (MB)

#Time	BPW%	AC	AvM	SCFS				
10:57:22	98	0	53	488	11:23:48	79	0	52	488
10:57:52	97	0	53	488	11:24:18	78	0	52	488
10:58:23	97	0	49	488	11:24:48	78	0	52	488
10:58:53	97	0	49	488	11:25:18	78	0	52	488
10:59:23	97	0	49	488	11:25:48	78	0	52	488
10:59:53	96	0	49	488	11:26:18	77	0	52	488
11:00:22	96	0	49	488	11:26:48	77	0	52	488
11:00:52	95	0	49	488	11:27:18	77	0	52	488
11:01:22	95	0	49	488	11:27:48	76	0	52	488
11:01:52	95	0	49	488	11:28:18	76	0	52	488
11:02:22	94	0	49	488	11:28:48	76	0	52	488
11:02:52	94	0	49	488	11:29:18	76	0	52	488
11:03:22	93	0	49	488	11:29:49	76	0	52	488
11:03:51	93	0	49	488	11:30:19	76	0	52	488
11:04:21	93	0	49	488	11:30:49	76	0	52	488
11:04:51	92	0	49	488	11:31:19	76	0	52	488
11:05:21	92	0	48	488	11:31:49	75	0	52	488
11:05:51	91	0	49	488	11:32:19	75	0	52	488
11:06:21	91	0	49	488	11:32:49	75	0	52	488
11:06:51	91	0	49	488	11:33:19	75	0	52	488
11:07:21	90	0	48	488	11:33:49	75	0	52	488
11:07:51	90	0	48	488	11:34:19	75	0	52	488
11:08:21	90	0	48	488	11:34:49	75	0	52	488
11:08:50	89	0	48	488	11:35:19	75	0	52	488
11:09:20	89	0	48	488	11:35:49	75	0	52	488
11:09:50	88	0	48	488	11:36:19	75	0	52	488
11:10:20	88	0	48	488	11:36:49	75	0	52	488
11:10:50	88	0	48	488	11:37:19	75	0	52	488
11:11:20	87	0	48	488	11:37:49	75	0	52	488
11:11:50	87	0	48	488	11:38:20	75	0	52	488
11:12:19	86	0	48	488	11:38:50	74	0	52	488
11:12:49	86	0	48	488	11:39:20	74	0	52	488
11:13:19	86	0	48	488	11:39:50	74	0	52	488
11:13:49	85	0	48	488	11:40:20	74	0	52	488
11:14:19	85	0	48	488	11:40:50	74	0	52	488
11:14:49	84	0	48	488	11:41:20	74	0	52	488
11:15:19	84	0	48	488	11:41:50	74	0	52	488
11:15:48	84	0	48	488	11:42:20	74	0	52	488
11:16:18	83	0	48	488	11:42:50	74	0	52	488
11:16:48	83	0	48	488	11:43:20	74	0	52	488
11:17:18	82	0	48	488	11:43:50	74	0	52	488
11:17:48	82	0	48	488	11:44:20	74	0	52	488
11:18:18	82	0	52	488	11:44:50	74	0	52	488
11:18:48	82	0	52	488	11:45:20	74	0	52	488
11:19:18	81	0	52	488	11:45:50	74	0	52	488
11:19:48	81	0	52	488	11:46:20	74	0	52	488
11:20:18	81	0	52	488	11:46:51	74	0	52	488
11:20:48	80	0	52	488	11:47:21	73	0	52	488
11:21:18	80	0	52	488	11:47:51	73	0	52	488
11:21:48	80	0	52	488	11:48:21	73	0	52	488
11:22:18	80	0	52	488	11:48:51	73	0	52	488
11:22:48	79	0	52	488	11:49:21	73	0	52	488
11:23:18	79	0	52	488	11:49:51	73	0	52	488

A.2 Status of the device during the WLAN ON – sleep 60 s test

#Information about Current Time, Battery Power %, AC Line Status, Available Memory (MB) and Storage Card Free Space (MB)

#Time	BPW%	AC	AvM	SCFS				
10:38:33	99	0	54	16	11:46:35	64	0	53	16
10:39:33	99	0	54	16	11:47:35	63	0	53	16
10:40:33	98	0	54	16	11:48:35	63	0	53	16
10:41:33	98	0	54	16	11:49:35	62	0	53	16
10:42:33	97	0	54	16	11:50:35	62	0	53	16
10:43:33	96	0	54	16	11:51:35	61	0	53	16
10:44:33	96	0	54	16	11:52:35	61	0	53	16
10:45:34	95	0	54	16	11:53:35	60	0	53	16
10:46:34	95	0	54	16	11:54:35	60	0	53	16
10:47:34	94	0	54	16	11:55:35	59	0	53	16
10:48:34	94	0	54	16	11:56:35	59	0	53	16
10:49:34	93	0	54	16	11:57:35	58	0	53	16
10:50:34	93	0	54	16	11:58:35	58	0	53	16
10:51:34	92	0	54	16	11:59:35	57	0	53	16
10:52:34	92	0	54	16	12:00:35	57	0	53	16
10:53:34	91	0	53	16	12:01:35	56	0	53	16
10:54:34	91	0	53	16	12:02:35	56	0	53	16
10:55:34	90	0	53	16	12:03:35	55	0	53	16
10:56:34	90	0	53	16	12:04:35	55	0	53	16
10:57:34	89	0	53	16	12:05:35	54	0	53	16
10:58:34	89	0	53	16	12:06:35	54	0	53	16
10:59:34	88	0	53	16	12:07:35	53	0	53	16
11:00:34	88	0	53	16	12:08:35	53	0	53	16
11:01:34	87	0	53	16	12:09:35	52	0	53	16
11:02:34	87	0	53	16	12:10:35	52	0	53	16
11:03:34	86	0	53	16	12:11:35	51	0	53	16
11:04:34	86	0	53	16	12:12:35	51	0	53	16
11:05:34	85	0	53	16	12:13:35	50	0	53	16
11:06:34	85	0	53	16	12:14:35	49	0	53	16
11:07:34	84	0	53	16	12:15:35	49	0	53	16
11:08:34	84	0	53	16	12:16:35	48	0	53	16
11:09:34	83	0	53	16	12:17:35	48	0	53	16
11:10:34	83	0	53	16	12:18:35	47	0	53	16
11:11:34	82	0	53	16	12:19:35	47	0	53	16
11:12:34	82	0	53	16	12:20:35	46	0	53	16
11:13:34	81	0	53	16	12:21:35	46	0	53	16
11:14:34	80	0	53	16	12:22:35	45	0	53	16
11:15:34	80	0	53	16	12:23:35	45	0	53	16
11:16:34	79	0	53	16	12:24:35	44	0	53	16
11:17:34	79	0	53	16	12:25:35	44	0	53	16
11:18:34	78	0	53	16	12:26:35	43	0	53	16
11:19:34	78	0	53	16	12:27:35	43	0	53	16
11:20:34	77	0	53	16	12:28:35	42	0	53	16
11:21:34	77	0	53	16	12:29:35	42	0	53	16
11:22:34	76	0	53	16	12:30:35	41	0	53	16
11:23:34	76	0	53	16	12:31:35	41	0	53	16
11:24:34	75	0	53	16	12:32:35	40	0	53	16
11:25:34	75	0	53	16	12:33:35	40	0	53	16
11:26:34	74	0	53	16	12:34:35	39	0	53	16
11:27:34	74	0	53	16	12:35:35	39	0	53	16
11:28:34	73	0	53	16	12:36:35	38	0	53	16
11:29:34	73	0	53	16	12:37:36	38	0	53	16
11:30:34	72	0	53	16	12:38:36	37	0	53	16
11:31:34	72	0	53	16	12:39:36	37	0	53	16
11:32:34	71	0	53	16	12:40:36	36	0	53	16
11:33:34	71	0	53	16	12:41:36	35	0	53	16
11:34:34	70	0	53	16	12:42:36	35	0	53	16
11:35:34	70	0	53	16	12:43:36	34	0	53	16
11:36:34	69	0	53	16	12:44:36	34	0	53	16
11:37:34	69	0	53	16	12:45:36	33	0	53	16
11:38:34	68	0	53	16	12:46:36	33	0	53	16
11:39:34	68	0	53	16	12:47:36	32	0	53	16
11:40:34	67	0	53	16	12:48:36	32	0	53	16
11:41:35	67	0	53	16	12:49:36	31	0	53	16
11:42:35	66	0	53	16	12:50:36	31	0	53	16
11:43:35	66	0	53	16	12:51:36	30	0	53	16
11:44:35	65	0	53	16	12:52:36	30	0	53	16
11:45:35	65	0	53	16	12:53:36	29	0	53	16
					12:54:36	29	0	53	16

12:55:36 28	0	53	16	14:10:40 9	0	53	16
12:56:36 28	0	53	16	14:11:40 8	0	53	16
12:57:36 27	0	53	16	14:12:40 8	0	53	16
12:58:36 27	0	53	16	14:13:40 8	0	53	16
12:59:36 26	0	53	16	14:14:40 8	0	53	16
13:00:36 26	0	53	16	14:15:40 8	0	53	16
13:01:36 25	0	53	16	14:16:40 8	0	53	16
13:02:36 25	0	53	16	14:17:40 8	0	53	16
13:03:36 24	0	53	16	14:18:40 8	0	53	16
13:04:36 24	0	53	16	14:19:40 8	0	53	16
13:05:36 23	0	53	16	14:20:40 7	0	53	16
13:06:36 23	0	53	16	14:21:40 7	0	53	16
13:07:36 22	0	53	16	14:22:40 7	0	53	16
13:08:36 21	0	53	16	14:23:40 7	0	53	16
13:09:36 21	0	53	16	14:24:40 7	0	53	16
13:10:36 20	0	53	16	14:25:40 7	0	53	16
13:11:36 20	0	53	16	14:26:41 7	0	53	16
13:12:36 19	0	53	16	14:27:41 7	0	53	16
13:13:36 19	0	53	16	14:28:41 6	0	53	16
13:14:36 18	0	53	16	14:29:41 6	0	53	16
13:15:36 18	0	53	16	14:30:41 6	0	53	16
13:16:36 17	0	53	16	14:31:41 6	0	53	16
13:17:36 17	0	53	16	14:32:41 6	0	53	16
13:18:36 16	0	53	16	14:33:41 6	0	53	16
13:19:36 16	0	53	16	14:34:41 6	0	53	16
13:20:36 15	0	53	16	14:35:41 6	0	53	16
13:21:36 15	0	53	16	14:36:41 6	0	53	16
13:22:36 14	0	53	16	14:37:41 5	0	53	16
13:23:36 14	0	53	16	14:38:41 5	0	53	16
13:24:36 14	0	53	16	14:39:41 5	0	53	16
13:25:37 14	0	53	16	14:40:41 5	0	53	16
13:26:37 14	0	53	16	14:41:42 5	0	53	16
13:27:37 14	0	53	16	14:42:42 5	0	53	16
13:28:37 14	0	53	16	14:43:42 5	0	53	16
13:29:37 13	0	53	16	14:44:42 5	0	53	16
13:30:37 13	0	53	16	14:45:42 5	0	53	16
13:31:37 13	0	53	16	14:46:42 4	0	53	16
13:32:37 13	0	53	16	14:47:42 4	0	53	16
13:33:37 13	0	53	16	14:48:42 4	0	53	16
13:34:37 13	0	53	16	14:49:42 4	0	53	16
13:35:37 13	0	53	16	14:50:42 4	0	53	16
13:36:37 13	0	53	16	14:51:42 4	0	53	16
13:37:37 12	0	53	16	14:52:42 4	0	53	16
13:38:37 12	0	53	16	14:53:42 4	0	53	16
13:39:38 12	0	53	16	14:54:42 4	0	53	16
13:40:38 12	0	53	16	14:55:42 3	0	53	16
13:41:38 12	0	53	16	14:56:42 3	0	53	16
13:42:38 12	0	53	16	14:57:43 3	0	53	16
13:43:38 12	0	53	16	14:58:43 3	0	53	16
13:44:38 12	0	53	16	14:59:43 3	0	53	16
13:45:38 11	0	53	16	15:00:43 3	0	53	16
13:46:38 11	0	53	16	15:01:43 3	0	53	16
13:47:38 11	0	53	16	15:02:43 3	0	53	16
13:48:38 11	0	53	16	15:03:43 3	0	53	16
13:49:38 11	0	53	16	15:04:43 2	0	53	16
13:50:38 11	0	53	16	15:05:43 2	0	53	16
13:51:38 11	0	53	16	15:06:43 2	0	53	16
13:52:38 11	0	53	16	15:07:43 2	0	53	16
13:53:38 10	0	53	16	15:08:43 2	0	53	16
13:54:39 10	0	53	16	15:09:43 2	0	53	16
13:55:39 10	0	53	16	15:10:43 2	0	53	16
13:56:39 10	0	53	16	15:11:43 2	0	53	16
13:57:39 10	0	53	16	15:12:43 2	0	53	16
13:58:39 10	0	53	16	15:13:44 1	0	53	16
13:59:39 10	0	53	16	15:14:44 1	0	53	16
14:00:39 10	0	53	16	15:15:44 1	0	53	16
14:01:39 10	0	53	16	15:16:44 1	0	53	16
14:02:39 9	0	53	16	15:17:44 1	0	53	16
14:03:39 9	0	53	16	15:18:44 1	0	53	16
14:04:39 9	0	53	16	15:19:44 1	0	53	16
14:05:39 9	0	53	16	15:20:44 1	0	53	16
14:06:39 9	0	53	16	15:21:44 1	0	53	16
14:07:39 9	0	53	16	15:22:44 0	0	53	16
14:08:39 9	0	53	16	15:23:44 0	0	53	16
14:09:39 9	0	53	16	15:24:44 0	0	53	16

```
15:25:44 0      0       53      16              15:28:45 0      0       53      16
15:26:44 0      0       53      16              15:29:45 0      0       53      16
15:27:45 0      0       53      16              15:30:45 0      0       53      16
```

A.3 Status of the device during the WLAN OFF – sleep 60 s test

```
#Information about Current Time, Battery Power %, AC Line Status, Available Memory (MB) and
Storage Card Free Space (MB)
#Time   BPW%    AC      AvM     SCFS            16:43:33 92     0       53      488
15:45:30 99     0       54      488             16:44:33 92     0       53      488
15:46:30 99     0       54      488             16:45:33 92     0       53      488
15:47:30 99     0       54      488             16:46:33 92     0       53      488
15:48:30 99     0       54      488             16:47:33 92     0       53      488
15:49:30 99     0       54      488             16:48:33 92     0       53      488
15:50:30 99     0       54      488             16:49:34 92     0       53      488
15:51:30 99     0       54      488             16:50:34 91     0       53      488
15:52:30 99     0       54      488             16:51:34 91     0       53      488
15:53:30 98     0       54      488             16:52:34 91     0       53      488
15:54:30 98     0       54      488             16:53:34 91     0       53      488
15:55:30 98     0       54      488             16:54:34 91     0       53      488
15:56:30 98     0       54      488             16:55:34 91     0       53      488
15:57:31 98     0       54      488             16:56:34 91     0       53      488
15:58:31 98     0       54      488             16:57:34 91     0       53      488
15:59:31 98     0       54      488             16:58:34 91     0       53      488
16:00:31 98     0       54      488             16:59:34 90     0       53      488
16:01:31 97     0       54      488             17:00:34 90     0       53      488
16:02:31 97     0       54      488             17:01:34 90     0       53      488
16:03:31 97     0       54      488             17:02:34 90     0       53      488
16:04:31 97     0       54      488             17:03:34 90     0       53      488
16:05:31 97     0       54      488             17:04:34 90     0       53      488
16:06:31 97     0       54      488             17:05:35 90     0       53      488
16:07:31 97     0       54      488             17:06:35 90     0       53      488
16:08:31 97     0       54      488             17:07:35 89     0       53      488
16:09:31 96     0       54      488             17:08:35 89     0       53      488
16:10:31 96     0       54      488             17:09:35 89     0       53      488
16:11:31 96     0       54      488             17:10:35 89     0       53      488
16:12:31 96     0       54      488             17:11:35 89     0       53      488
16:13:31 96     0       53      488             17:12:35 89     0       53      488
16:14:31 96     0       53      488             17:13:35 89     0       53      488
16:15:32 96     0       53      488             17:14:35 89     0       53      488
16:16:32 96     0       53      488             17:15:35 88     0       53      488
16:17:32 96     0       53      488             17:16:35 88     0       53      488
16:18:32 95     0       53      488             17:17:35 88     0       53      488
16:19:32 95     0       53      488             17:18:35 88     0       53      488
16:20:32 95     0       53      488             17:19:35 88     0       53      488
16:21:32 95     0       53      488             17:20:35 88     0       53      488
16:22:32 95     0       53      488             17:21:35 88     0       53      488
16:23:32 95     0       53      488             17:22:36 88     0       53      488
16:24:32 95     0       53      488             17:23:36 88     0       53      488
16:25:32 95     0       53      488             17:24:36 87     0       53      488
16:26:32 94     0       53      488             17:25:36 87     0       53      488
16:27:32 94     0       53      488             17:26:36 87     0       53      488
16:28:32 94     0       53      488             17:27:36 87     0       53      488
16:29:32 94     0       53      488             17:28:36 87     0       53      488
16:30:32 94     0       53      488             17:29:36 87     0       53      488
16:31:33 94     0       53      488             17:30:36 87     0       53      488
16:32:33 94     0       53      488             17:31:36 86     0       53      488
16:33:33 94     0       53      488             17:32:36 86     0       53      488
16:34:33 93     0       53      488             17:33:36 86     0       53      488
16:35:33 93     0       53      488             17:34:36 86     0       53      488
16:36:33 93     0       53      488             17:35:36 86     0       53      488
16:37:33 93     0       53      488             17:36:36 86     0       53      488
16:38:33 93     0       53      488             17:37:36 86     0       53      488
16:39:33 93     0       53      488             17:38:36 86     0       53      488
16:40:33 93     0       53      488             17:39:37 86     0       53      488
16:41:33 93     0       53      488             17:40:37 85     0       53      488
16:42:33 93     0       53      488             17:41:37 85     0       53      488
```

17:42:37 85	0	53	488	18:57:41 76	0	53	488
17:43:37 85	0	53	488	18:58:41 76	0	53	488
17:44:37 85	0	53	488	18:59:42 76	0	53	488
17:45:37 85	0	53	488	19:00:42 76	0	53	488
17:46:37 85	0	53	488	19:01:42 75	0	53	488
17:47:37 85	0	53	488	19:02:42 75	0	53	488
17:48:37 84	0	53	488	19:03:42 75	0	53	488
17:49:37 84	0	53	488	19:04:42 75	0	53	488
17:50:37 84	0	53	488	19:05:42 75	0	53	488
17:51:37 84	0	53	488	19:06:42 75	0	53	488
17:52:37 84	0	53	488	19:07:42 75	0	53	488
17:53:37 84	0	53	488	19:08:42 75	0	53	488
17:54:37 84	0	53	488	19:09:42 74	0	53	488
17:55:38 84	0	53	488	19:10:42 74	0	53	488
17:56:38 83	0	53	488	19:11:42 74	0	53	488
17:57:38 83	0	53	488	19:12:42 74	0	53	488
17:58:38 83	0	53	488	19:13:42 74	0	53	488
17:59:38 83	0	53	488	19:14:42 74	0	53	488
18:00:38 83	0	53	488	19:15:42 74	0	53	488
18:01:38 83	0	53	488	19:16:42 74	0	53	488
18:02:38 83	0	53	488	19:17:43 73	0	53	488
18:03:38 83	0	53	488	19:18:43 73	0	53	488
18:04:38 82	0	53	488	19:19:43 73	0	53	488
18:05:38 82	0	53	488	19:20:43 73	0	53	488
18:06:38 82	0	53	488	19:21:43 73	0	53	488
18:07:38 82	0	53	488	19:22:43 73	0	53	488
18:08:38 82	0	53	488	19:23:43 73	0	53	488
18:09:38 82	0	53	488	19:24:43 73	0	53	488
18:10:38 82	0	53	488	19:25:43 72	0	53	488
18:11:39 82	0	53	488	19:26:43 72	0	53	488
18:12:39 82	0	53	488	19:27:43 72	0	53	488
18:13:39 81	0	53	488	19:28:43 72	0	53	488
18:14:39 81	0	53	488	19:29:43 72	0	53	488
18:15:39 81	0	53	488	19:30:43 72	0	53	488
18:16:39 81	0	53	488	19:31:43 72	0	53	488
18:17:39 81	0	53	488	19:32:43 72	0	53	488
18:18:39 81	0	53	488	19:33:43 71	0	53	488
18:19:39 81	0	53	488	19:34:43 71	0	53	488
18:20:39 80	0	53	488	19:35:44 71	0	53	488
18:21:39 80	0	53	488	19:36:44 71	0	53	488
18:22:39 80	0	53	488	19:37:44 71	0	53	488
18:23:39 80	0	53	488	19:38:44 71	0	53	488
18:24:39 80	0	53	488	19:39:44 71	0	53	488
18:25:39 80	0	53	488	19:40:44 71	0	53	488
18:26:40 80	0	53	488	19:41:44 70	0	53	488
18:27:40 80	0	53	488	19:42:44 70	0	53	488
18:28:40 80	0	53	488	19:43:44 70	0	53	488
18:29:40 79	0	53	488	19:44:44 70	0	53	488
18:30:40 79	0	53	488	19:45:44 70	0	53	488
18:31:40 79	0	53	488	19:46:44 70	0	53	488
18:32:40 79	0	53	488	19:47:44 70	0	53	488
18:33:40 79	0	53	488	19:48:44 70	0	53	488
18:34:40 79	0	53	488	19:49:44 69	0	53	488
18:35:40 79	0	53	488	19:50:44 69	0	53	488
18:36:40 79	0	53	488	19:51:44 69	0	53	488
18:37:40 78	0	53	488	19:52:45 69	0	53	488
18:38:40 78	0	53	488	19:53:45 69	0	53	488
18:39:40 78	0	53	488	19:54:45 69	0	53	488
18:40:40 78	0	53	488	19:55:45 69	0	53	488
18:41:40 78	0	53	488	19:56:45 69	0	53	488
18:42:41 78	0	53	488	19:57:45 68	0	53	488
18:43:41 78	0	53	488	19:58:45 68	0	53	488
18:44:41 77	0	53	488	19:59:45 68	0	53	488
18:45:41 77	0	53	488	20:00:45 68	0	53	488
18:46:41 77	0	53	488	20:01:45 68	0	53	488
18:47:41 77	0	53	488	20:02:45 68	0	53	488
18:48:41 77	0	53	488	20:03:45 68	0	53	488
18:49:41 77	0	53	488	20:04:45 68	0	53	488
18:50:41 77	0	53	488	20:05:45 67	0	53	488
18:51:41 77	0	53	488	20:06:45 67	0	53	488
18:52:41 77	0	53	488	20:07:45 67	0	53	488
18:53:41 76	0	53	488	20:08:45 67	0	53	488
18:54:41 76	0	53	488	20:09:46 67	0	53	488
18:55:41 76	0	53	488	20:10:46 67	0	53	488
18:56:41 76	0	53	488	20:11:46 67	0	53	488

20:12:46 67	0	53	488	21:27:50 57	0	53	488
20:13:46 66	0	53	488	21:28:50 57	0	53	488
20:14:46 66	0	53	488	21:29:50 57	0	53	488
20:15:46 66	0	53	488	21:30:50 57	0	53	488
20:16:46 66	0	53	488	21:31:51 57	0	53	488
20:17:46 66	0	53	488	21:32:51 56	0	53	488
20:18:46 66	0	53	488	21:33:51 56	0	53	488
20:19:46 66	0	53	488	21:34:51 56	0	53	488
20:20:46 66	0	53	488	21:35:51 56	0	53	488
20:21:46 65	0	53	488	21:36:51 56	0	53	488
20:22:46 65	0	53	488	21:37:51 56	0	53	488
20:23:46 65	0	53	488	21:38:51 56	0	53	488
20:24:46 65	0	53	488	21:39:51 56	0	53	488
20:25:47 65	0	53	488	21:40:51 55	0	53	488
20:26:47 65	0	53	488	21:41:51 55	0	53	488
20:27:47 65	0	53	488	21:42:51 55	0	53	488
20:28:47 65	0	53	488	21:43:51 55	0	53	488
20:29:47 64	0	53	488	21:44:51 55	0	53	488
20:30:47 64	0	53	488	21:45:51 55	0	53	488
20:31:47 64	0	53	488	21:46:51 55	0	53	488
20:32:47 64	0	53	488	21:47:52 55	0	53	488
20:33:47 64	0	53	488	21:48:52 54	0	53	488
20:34:47 64	0	53	488	21:49:52 54	0	53	488
20:35:47 64	0	53	488	21:50:52 54	0	53	488
20:36:47 64	0	53	488	21:51:52 54	0	53	488
20:37:47 63	0	53	488	21:52:52 54	0	53	488
20:38:47 63	0	53	488	21:53:52 54	0	53	488
20:39:47 63	0	53	488	21:54:52 54	0	53	488
20:40:47 63	0	53	488	21:55:52 54	0	53	488
20:41:47 63	0	53	488	21:56:52 53	0	53	488
20:42:48 63	0	53	488	21:57:52 53	0	53	488
20:43:48 63	0	53	488	21:58:52 53	0	53	488
20:44:48 62	0	53	488	21:59:52 53	0	53	488
20:45:48 62	0	53	488	22:00:52 53	0	53	488
20:46:48 62	0	53	488	22:01:52 53	0	53	488
20:47:48 62	0	53	488	22:02:52 53	0	53	488
20:48:48 62	0	53	488	22:03:52 53	0	53	488
20:49:48 62	0	53	488	22:04:52 52	0	53	488
20:50:48 62	0	53	488	22:05:52 52	0	53	488
20:51:48 62	0	53	488	22:06:52 52	0	53	488
20:52:48 62	0	53	488	22:07:53 52	0	53	488
20:53:48 61	0	53	488	22:08:53 52	0	53	488
20:54:48 61	0	53	488	22:09:53 52	0	53	488
20:55:48 61	0	53	488	22:10:53 52	0	53	488
20:56:48 61	0	53	488	22:11:53 52	0	53	488
20:57:48 61	0	53	488	22:12:53 51	0	53	488
20:58:49 61	0	53	488	22:13:53 51	0	53	488
20:59:49 61	0	53	488	22:14:53 51	0	53	488
21:00:49 61	0	53	488	22:15:53 51	0	53	488
21:01:49 60	0	53	488	22:16:53 51	0	53	488
21:02:49 60	0	53	488	22:17:53 51	0	53	488
21:03:49 60	0	53	488	22:18:53 51	0	53	488
21:04:49 60	0	53	488	22:19:53 51	0	53	488
21:05:49 60	0	53	488	22:20:53 50	0	53	488
21:06:49 60	0	53	488	22:21:53 50	0	53	488
21:07:49 60	0	53	488	22:22:53 50	0	53	488
21:08:49 59	0	53	488	22:23:53 50	0	53	488
21:09:49 59	0	53	488	22:24:54 50	0	53	488
21:10:49 59	0	53	488	22:25:54 50	0	53	488
21:11:49 59	0	53	488	22:26:54 50	0	53	488
21:12:49 59	0	53	488	22:27:54 50	0	53	488
21:13:49 59	0	53	488	22:28:54 49	0	53	488
21:14:50 59	0	53	488	22:29:54 49	0	53	488
21:15:50 59	0	53	488	22:30:54 49	0	53	488
21:16:50 59	0	53	488	22:31:54 49	0	53	488
21:17:50 58	0	53	488	22:32:54 49	0	53	488
21:18:50 58	0	53	488	22:33:54 49	0	53	488
21:19:50 58	0	53	488	22:34:54 49	0	53	488
21:20:50 58	0	53	488	22:35:54 48	0	53	488
21:21:50 58	0	53	488	22:36:54 48	0	53	488
21:22:50 58	0	53	488	22:37:54 48	0	53	488
21:23:50 58	0	53	488	22:38:54 48	0	53	488
21:24:50 58	0	53	488	22:39:54 48	0	53	488
21:25:50 57	0	53	488	22:40:54 48	0	53	488
21:26:50 57	0	53	488	22:41:54 48	0	53	488

22:42:54 48	0	53	488	23:57:59 38	0	53	488
22:43:54 48	0	53	488	23:58:59 38	0	53	488
22:44:55 47	0	53	488	23:59:59 38	0	53	488
22:45:55 47	0	53	488	00:00:59 38	0	53	488
22:46:55 47	0	53	488	00:01:59 37	0	53	488
22:47:55 47	0	53	488	00:02:59 37	0	53	488
22:48:55 47	0	53	488	00:03:59 37	0	53	488
22:49:55 47	0	53	488	00:04:59 37	0	53	488
22:50:55 47	0	53	488	00:05:59 37	0	53	488
22:51:55 46	0	53	488	00:06:59 37	0	53	488
22:52:55 46	0	53	488	00:07:59 37	0	53	488
22:53:55 46	0	53	488	00:08:59 37	0	53	488
22:54:55 46	0	53	488	00:09:59 36	0	53	488
22:55:55 46	0	53	488	00:10:59 36	0	53	488
22:56:55 46	0	53	488	00:11:59 36	0	53	488
22:57:55 46	0	53	488	00:13:00 36	0	53	488
22:58:55 46	0	53	488	00:14:00 36	0	53	488
22:59:55 45	0	53	488	00:15:00 36	0	53	488
23:00:55 45	0	53	488	00:16:00 36	0	53	488
23:01:56 45	0	53	488	00:17:00 36	0	53	488
23:02:56 45	0	53	488	00:18:00 35	0	53	488
23:03:56 45	0	53	488	00:19:00 35	0	53	488
23:04:56 45	0	53	488	00:20:00 35	0	53	488
23:05:56 45	0	53	488	00:21:00 35	0	53	488
23:06:56 45	0	53	488	00:22:00 35	0	53	488
23:07:56 44	0	53	488	00:23:00 35	0	53	488
23:08:56 44	0	53	488	00:24:00 35	0	53	488
23:09:56 44	0	53	488	00:25:00 35	0	53	488
23:10:56 44	0	53	488	00:26:00 34	0	53	488
23:11:56 44	0	53	488	00:27:00 34	0	53	488
23:12:56 44	0	53	488	00:28:01 34	0	53	488
23:13:56 44	0	53	488	00:29:01 34	0	53	488
23:14:56 44	0	53	488	00:30:01 34	0	53	488
23:15:56 43	0	53	488	00:31:01 34	0	53	488
23:16:56 43	0	53	488	00:32:01 34	0	53	488
23:17:56 43	0	53	488	00:33:01 33	0	53	488
23:18:56 43	0	53	488	00:34:01 33	0	53	488
23:19:56 43	0	53	488	00:35:01 33	0	53	488
23:20:57 43	0	53	488	00:36:01 33	0	53	488
23:21:57 43	0	53	488	00:37:01 33	0	53	488
23:22:57 43	0	53	488	00:38:01 33	0	53	488
23:23:57 42	0	53	488	00:39:01 33	0	53	488
23:24:57 42	0	53	488	00:40:01 33	0	53	488
23:25:57 42	0	53	488	00:41:01 32	0	53	488
23:26:57 42	0	53	488	00:42:01 32	0	53	488
23:27:57 42	0	53	488	00:43:01 32	0	53	488
23:28:57 42	0	53	488	00:44:01 32	0	53	488
23:29:57 42	0	53	488	00:45:02 32	0	53	488
23:30:57 41	0	53	488	00:46:02 32	0	53	488
23:31:57 41	0	53	488	00:47:02 32	0	53	488
23:32:57 41	0	53	488	00:48:02 32	0	53	488
23:33:57 41	0	53	488	00:49:02 31	0	53	488
23:34:57 41	0	53	488	00:50:02 31	0	53	488
23:35:57 41	0	53	488	00:51:02 31	0	53	488
23:36:57 41	0	53	488	00:52:02 31	0	53	488
23:37:58 41	0	53	488	00:53:02 31	0	53	488
23:38:58 40	0	53	488	00:54:02 31	0	53	488
23:39:58 40	0	53	488	00:55:02 31	0	53	488
23:40:58 40	0	53	488	00:56:02 30	0	53	488
23:41:58 40	0	53	488	00:57:02 30	0	53	488
23:42:58 40	0	53	488	00:58:02 30	0	53	488
23:43:58 40	0	53	488	00:59:02 30	0	53	488
23:44:58 40	0	53	488	01:00:02 30	0	53	488
23:45:58 40	0	53	488	01:01:03 30	0	53	488
23:46:58 39	0	53	488	01:02:03 30	0	53	488
23:47:58 39	0	53	488	01:03:03 30	0	53	488
23:48:58 39	0	53	488	01:04:03 29	0	53	488
23:49:58 39	0	53	488	01:05:03 29	0	53	488
23:50:58 39	0	53	488	01:06:03 29	0	53	488
23:51:58 39	0	53	488	01:07:03 29	0	53	488
23:52:58 39	0	53	488	01:08:03 29	0	53	488
23:53:58 38	0	53	488	01:09:03 29	0	53	488
23:54:59 38	0	53	488	01:10:03 29	0	53	488
23:55:59 38	0	53	488	01:11:03 29	0	53	488
23:56:59 38	0	53	488	01:12:03 28	0	53	488

01:13:03 28	0	53	488	02:28:07 19	0	53	488
01:14:03 28	0	53	488	02:29:07 18	0	53	488
01:15:03 28	0	53	488	02:30:07 18	0	53	488
01:16:03 28	0	53	488	02:31:07 18	0	53	488
01:17:03 28	0	53	488	02:32:07 18	0	53	488
01:18:03 28	0	53	488	02:33:07 18	0	53	488
01:19:03 28	0	53	488	02:34:07 18	0	53	488
01:20:03 27	0	53	488	02:35:07 18	0	53	488
01:21:04 27	0	53	488	02:36:08 17	0	53	488
01:22:04 27	0	53	488	02:37:08 17	0	53	488
01:23:04 27	0	53	488	02:38:08 17	0	53	488
01:24:04 27	0	53	488	02:39:08 17	0	53	488
01:25:04 27	0	53	488	02:40:08 17	0	53	488
01:26:04 27	0	53	488	02:41:08 17	0	53	488
01:27:04 26	0	53	488	02:42:08 17	0	53	488
01:28:04 26	0	53	488	02:43:08 17	0	53	488
01:29:04 26	0	53	488	02:44:08 16	0	53	488
01:30:04 26	0	53	488	02:45:08 16	0	53	488
01:31:04 26	0	53	488	02:46:08 16	0	53	488
01:32:04 26	0	53	488	02:47:08 16	0	53	488
01:33:04 26	0	53	488	02:48:08 16	0	53	488
01:34:04 26	0	53	488	02:49:08 16	0	53	488
01:35:04 25	0	53	488	02:50:08 16	0	53	488
01:36:04 25	0	53	488	02:51:08 16	0	53	488
01:37:04 25	0	53	488	02:52:08 15	0	53	488
01:38:04 25	0	53	488	02:53:09 15	0	53	488
01:39:05 25	0	53	488	02:54:09 15	0	53	488
01:40:05 25	0	53	488	02:55:09 15	0	53	488
01:41:05 25	0	53	488	02:56:09 15	0	53	488
01:42:05 25	0	53	488	02:57:09 15	0	53	488
01:43:05 24	0	53	488	02:58:09 15	0	53	488
01:44:05 24	0	53	488	02:59:09 14	0	53	488
01:45:05 24	0	53	488	03:00:09 14	0	53	488
01:46:05 24	0	53	488	03:01:09 14	0	53	488
01:47:05 24	0	53	488	03:02:09 14	0	53	488
01:48:05 24	0	53	488	03:03:09 14	0	53	488
01:49:05 24	0	53	488	03:04:09 14	0	53	488
01:50:05 24	0	53	488	03:05:09 14	0	53	488
01:51:05 23	0	53	488	03:06:09 14	0	53	488
01:52:05 23	0	53	488	03:07:09 13	0	53	488
01:53:05 23	0	53	488	03:08:09 13	0	53	488
01:54:05 23	0	53	488	03:09:09 13	0	53	488
01:55:05 23	0	53	488	03:10:09 13	0	53	488
01:56:05 23	0	53	488	03:11:10 13	0	53	488
01:57:05 23	0	53	488	03:12:10 13	0	53	488
01:58:05 22	0	53	488	03:13:10 13	0	53	488
01:59:06 22	0	53	488	03:14:10 12	0	53	488
02:00:06 22	0	53	488	03:15:10 12	0	53	488
02:01:06 22	0	53	488	03:16:10 12	0	53	488
02:02:06 22	0	53	488	03:17:10 12	0	53	488
02:03:06 22	0	53	488	03:18:10 12	0	53	488
02:04:06 22	0	53	488	03:19:10 12	0	53	488
02:05:06 22	0	53	488	03:20:10 12	0	53	488
02:06:06 21	0	53	488	03:21:10 12	0	53	488
02:07:06 21	0	53	488	03:22:10 11	0	53	488
02:08:06 21	0	53	488	03:23:10 11	0	53	488
02:09:06 21	0	53	488	03:24:10 11	0	53	488
02:10:06 21	0	53	488	03:25:10 11	0	53	488
02:11:06 21	0	53	488	03:26:10 11	0	53	488
02:12:06 21	0	53	488	03:27:10 11	0	53	488
02:13:06 20	0	53	488	03:28:10 11	0	53	488
02:14:06 20	0	53	488	03:29:11 11	0	53	488
02:15:06 20	0	53	488	03:30:11 10	0	53	488
02:16:06 20	0	53	488	03:31:11 10	0	53	488
02:17:07 20	0	53	488	03:32:11 10	0	53	488
02:18:07 20	0	53	488	03:33:11 10	0	53	488
02:19:07 20	0	53	488	03:34:11 10	0	53	488
02:20:07 20	0	53	488	03:35:11 10	0	53	488
02:21:07 19	0	53	488	03:36:11 10	0	53	488
02:22:07 19	0	53	488	03:37:11 9	0	53	488
02:23:07 19	0	53	488	03:38:11 9	0	53	488
02:24:07 19	0	53	488	03:39:11 9	0	53	488
02:25:07 19	0	53	488	03:40:11 9	0	53	488
02:26:07 19	0	53	488	03:41:11 9	0	53	488
02:27:07 19	0	53	488	03:42:11 9	0	53	488

03:43:11 9	0	53	488	04:18:14 4	0	53	488
03:44:11 9	0	53	488	04:19:14 4	0	53	488
03:45:12 8	0	53	488	04:20:14 4	0	53	488
03:46:12 8	0	53	488	04:21:14 4	0	53	488
03:47:12 8	0	53	488	04:22:14 4	0	53	488
03:48:12 8	0	53	488	04:23:14 3	0	53	488
03:49:12 8	0	53	488	04:24:14 3	0	53	488
03:50:12 8	0	53	488	04:25:14 3	0	53	488
03:51:12 8	0	53	488	04:26:14 3	0	53	488
03:52:12 8	0	53	488	04:27:14 3	0	53	488
03:53:12 7	0	53	488	04:28:14 3	0	53	488
03:54:12 7	0	53	488	04:29:14 3	0	53	488
03:55:12 7	0	53	488	04:30:14 3	0	53	488
03:56:12 7	0	53	488	04:31:14 2	0	53	488
03:57:12 7	0	53	488	04:32:14 2	0	53	488
03:58:12 7	0	53	488	04:33:14 2	0	53	488
03:59:12 7	0	53	488	04:34:14 2	0	53	488
04:00:12 6	0	53	488	04:35:14 2	0	53	488
04:01:13 6	0	53	488	04:36:14 2	0	53	488
04:02:13 6	0	53	488	04:37:15 2	0	53	488
04:03:13 6	0	53	488	04:38:15 1	0	53	488
04:04:13 6	0	53	488	04:39:15 1	0	53	488
04:05:13 6	0	53	488	04:40:15 1	0	53	488
04:06:13 6	0	53	488	04:41:15 1	0	53	488
04:07:13 6	0	53	488	04:42:15 1	0	53	488
04:08:13 5	0	53	488	04:43:15 1	0	53	488
04:09:13 5	0	53	488	04:44:15 1	0	53	488
04:10:13 5	0	53	488	04:45:15 1	0	53	488
04:11:13 5	0	53	488	04:46:15 0	0	53	488
04:12:13 5	0	53	488	04:47:15 0	0	53	488
04:13:13 5	0	53	488	04:48:15 0	0	53	488
04:14:13 5	0	53	488	04:49:15 0	0	53	488
04:15:13 4	0	53	488	04:50:15 0	0	53	488
04:16:13 4	0	53	488	04:51:15 0	0	53	488
04:17:13 4	0	53	488	04:52:15 0	0	53	488

A.4 Status Client application Source Code

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Runtime.InteropServices;
using System.Threading;
using System.Text;
using System.IO;
using OpenNETCF.IO;


// ***** From here to line 203 is the OpenNETCF Namespace *****

//====================================================================================
//
//      OpenNETCF.IO.StorageCard
//      Copyright (C) 2003-2004, OpenNETCF.org
//
//      This library is free software; you can redistribute it and/or modify it under
//      the terms of the OpenNETCF.org Shared Source License.
//
//      This library is distributed in the hope that it will be useful, but
//      WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
//      FITNESS FOR A PARTICULAR PURPOSE. See the OpenNETCF.org Shared Source License
//      for more details.
//
//      You should have received a copy of the OpenNETCF.org Shared Source License
//      along with this library; if not, email licensing@opennetcf.org to request a copy.
//
//      If you wish to contact the OpenNETCF Advisory Board to discuss licensing, please
//      email licensing@opennetcf.org.
//
//      For general enquiries, email enquiries@opennetcf.org or visit our website at:
//      http://www.opennetcf.org
//
//====================================================================================


namespace OpenNETCF.IO
{
        #region Storage Card
        /// <summary>
        /// Provides access to removable storage cards on the device
        /// </summary>
        public class StorageCard
        {
                private StorageCard(){}

                //storage cards are directories with the temporary attribute
                private const System.IO.FileAttributes attrStorageCard =
System.IO.FileAttributes.Directory | System.IO.FileAttributes.Temporary;

                #region Get StorageCard Names
                // Based on original sample code by Alex Feinman:-
                // http://www.opennetcf.org/forums/topic.asp?TOPIC_ID=432

                /// <summary>
                /// Returns the Storage Cards currently attached to the device
                /// </summary>
                /// <returns>Array containing the names of Storage Cards currently attached to
the device.</returns>
                public static string[] GetStorageCardNames()
                {
                        ArrayList scards = new ArrayList();

                        DirectoryInfo rootDir = new DirectoryInfo(@"\");
```

```
                        foreach(DirectoryInfo di in rootDir.GetDirectories() )
                        {
                                //if directory and temporary
                                if ( (di.Attributes & attrStorageCard) == attrStorageCard )
                                {
                                        //add to collection of storage cards
                                        scards.Add(di.Name);
                                }
                        }
                        return (string[])scards.ToArray(typeof(string));
                }
                #endregion

                #region Get StorageCards
                /// <summary>
                /// Returns an array of <see cref="T:System.IO.DirectoryInfo"/> entries listing
all the Storage Cards currently attached
                /// </summary>
                /// <returns>An array of <see cref="T:System.IO.DirectoryInfo"/>
entries.</returns>
                public static DirectoryInfo[] GetStorageCards()
                {
                        ArrayList scards = new ArrayList();

                        DirectoryInfo rootDir = new DirectoryInfo(@"\");

                        foreach( DirectoryInfo di in rootDir.GetDirectories() )
                        {
                                if ( (di.Attributes & attrStorageCard) == attrStorageCard )
                                {
                                        scards.Add(di);
                                }
                        }
                        return
(DirectoryInfo[])scards.ToArray(Type.GetType("System.IO.DirectoryInfo"));
                }
                #endregion

                #region Get Documents Folder
                /*/// <summary>
                /// Gets the path of the user documents folder on the specified storage card
                /// </summary>
                /// <param name="storageCard">Path to storage card e.g. "\Storage Card"</param>
                /// <returns>Path to user documents folder e.g. "\Storage Card\My
Documents"</returns>
                public static string GetDocumentsFolder(string storageCard)
                {
                        //create buffer (Max Path)
                        char[] buffer = new char[256];

                        //get documents folder for supplied storage card
                        bool success = SHGetDocumentsFolder(storageCard, ref buffer);

                        if(!success)
                        {

        Console.WriteLine(System.Runtime.InteropServices.Marshal.GetLastWin32Error().ToString());
                        }

                        //return documents folder as string
                        return new string(buffer).TrimEnd('\0');
                }

                [DllImport("ceshell.dll", EntryPoint="#75", SetLastError=true)]
                private static extern bool SHGetDocumentsFolder(string pszVolume, ref char[]
pszDocs);
                */
                #endregion

                #region GetFreeSpace
                /// <summary>
                /// Obtains the following information about the amount of space available on a
disk volume: the total amount of space, the total amount of free space, and the amount of free
space available to the user associated with the calling thread.
                /// <para><b>New in v1.1</b></para>
```

```
		/// </summary>
		/// <param name="directoryName"><see cref="System.String"/> that specifies a
directory on the specified disk.
		/// This string can be a Universal Naming Convention (UNC) name.</param>
		/// <returns>A <see cref="DiskFreeSpace"/> structure containing details about the
space available on the specified storage media.</returns>
		public static DiskFreeSpace GetDiskFreeSpace(string directoryName)
		{
			DiskFreeSpace result = new DiskFreeSpace();

			if(!GetDiskFreeSpaceEx(directoryName, ref result.FreeBytesAvailable, ref
result.TotalBytes, ref result.TotalFreeBytes))
			{
				throw new
System.ComponentModel.Win32Exception(Marshal.GetLastWin32Error(), "Error retrieving free disk
space");
			}

			return result;
		}

		/// <summary>
		/// Describes the free space on a storage card.
		/// <para><b>New in v1.1</b></para>
		/// </summary>
		public struct DiskFreeSpace
		{
			/// <summary>
			/// The total number of free bytes on the disk that are available to the
user associated with the calling thread.
			/// </summary>
			public long FreeBytesAvailable;
			/// <summary>
			/// The total number of bytes on the disk that are available to the user
associated with the calling thread.
			/// </summary>
			public long TotalBytes;
			/// <summary>
			/// The total number of free bytes on the disk.
			/// </summary>
			public long TotalFreeBytes;
		}

		[DllImport("coredll")]
		private static extern bool GetDiskFreeSpaceEx(string directoryName, ref long
freeBytesAvailable, ref long totalBytes, ref long totalFreeBytes);

		#endregion

		#region AutoRun Path
		/// <summary>
		/// Returns the path Windows CE will use to search for an AutoRun application
when a Storage Card is inserted
		/// </summary>
		/// <remarks>Requires a Pocket PC or Windows CE.NET 4.2 with AYGShell
extensions.</remarks>
		public static string AutoRunPath
		{
			get
			{
				//create buffer (Max Path)
				char[] buffer = new char[256];

				//get documents folder for supplied storage card
				bool result = SHGetAutoRunPath(buffer);

				//return documents folder as string
				return new String(buffer).TrimEnd('\0');
			}
		}

		[DllImport("aygshell.dll", EntryPoint="SHGetAutoRunPath", SetLastError=true)]
		private static extern bool SHGetAutoRunPath(char[]  pAutoRunPath );

		#endregion
```

```
        }
        #endregion
}

//=============================================================================



namespace STATUS_CLIENT
{
        /// <summary>
        /// Summary description for Form1.
        /// </summary>
        public class Form1 : System.Windows.Forms.Form
        {
                private UdpClient client = new UdpClient();
                private int port = 11112;
                private bool loop = true;
                //
                private System.Windows.Forms.Label lblIpAddr;
                private System.Windows.Forms.Label lblSleep;
                private System.Windows.Forms.Button btnStart;
                private System.Windows.Forms.CheckBox chkStore;
                private System.Windows.Forms.CheckBox chkSend;
                private System.Windows.Forms.TextBox txtIpAddr;
                private System.Windows.Forms.TextBox txtSleep;
                private System.Windows.Forms.MainMenu mainMenu1;

                public Form1()
                {
                        //
                        // Required for Windows Form Designer support
                        //
                        InitializeComponent();

                }
                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
                protected override void Dispose( bool disposing )
                {
                        base.Dispose( disposing );
                }
                #region Windows Form Designer generated code
                /// <summary>
                /// Required method for Designer support - do not modify
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                        this.lblIpAddr = new System.Windows.Forms.Label();
                        this.lblSleep = new System.Windows.Forms.Label();
                        this.btnStart = new System.Windows.Forms.Button();
                        this.chkStore = new System.Windows.Forms.CheckBox();
                        this.chkSend = new System.Windows.Forms.CheckBox();
                        this.txtIpAddr = new System.Windows.Forms.TextBox();
                        this.txtSleep = new System.Windows.Forms.TextBox();
                        this.mainMenu1 = new System.Windows.Forms.MainMenu();
                        //
                        // lblIpAddr
                        //
                        this.lblIpAddr.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.lblIpAddr.Location = new System.Drawing.Point(8, 66);
                        this.lblIpAddr.Size = new System.Drawing.Size(96, 16);
                        this.lblIpAddr.Text = "IP Address :";
                        this.lblIpAddr.TextAlign = System.Drawing.ContentAlignment.TopCenter;
                        //
                        // lblSleep
                        //
                        this.lblSleep.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.lblSleep.Location = new System.Drawing.Point(8, 114);
```

```
                        this.lblSleep.Size = new System.Drawing.Size(176, 24);
                        this.lblSleep.Text = "Sleep Time (milliseconds) :";
                        //
                        // btnStart
                        //
                        this.btnStart.Enabled = false;
                        this.btnStart.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.btnStart.Location = new System.Drawing.Point(8, 144);
                        this.btnStart.Size = new System.Drawing.Size(224, 32);
                        this.btnStart.Text = "START";
                        this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
                        //
                        // chkStore
                        //
                        this.chkStore.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.chkStore.Location = new System.Drawing.Point(8, 8);
                        this.chkStore.Size = new System.Drawing.Size(224, 16);
                        this.chkStore.Text = " Store data in a file";
                        //
                        // chkSend
                        //
                        this.chkSend.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.chkSend.Location = new System.Drawing.Point(8, 32);
                        this.chkSend.Size = new System.Drawing.Size(224, 16);
                        this.chkSend.Text = " Send data to a remote machine";
                        //
                        // txtIpAddr
                        //
                        this.txtIpAddr.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.txtIpAddr.Location = new System.Drawing.Point(104, 64);
                        this.txtIpAddr.Size = new System.Drawing.Size(128, 22);
                        this.txtIpAddr.Text = "0.0.0.0";
                        //
                        // txtSleep
                        //
                        this.txtSleep.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.txtSleep.Location = new System.Drawing.Point(192, 112);
                        this.txtSleep.Size = new System.Drawing.Size(40, 22);
                        this.txtSleep.Text = "";
                        this.txtSleep.TextChanged += new
System.EventHandler(this.txtSleep_TextChanged);
                        //
                        // Form1
                        //
                        this.Controls.Add(this.chkStore);
                        this.Controls.Add(this.btnStart);
                        this.Controls.Add(this.lblIpAddr);
                        this.Controls.Add(this.lblSleep);
                        this.Controls.Add(this.chkSend);
                        this.Controls.Add(this.txtIpAddr);
                        this.Controls.Add(this.txtSleep);
                        this.Menu = this.mainMenu1;
                        this.Text = "Status Client";

                }
                #endregion

                /// <summary>
                /// The main entry point for the application.
                /// </summary>


                public class SYSTEM_POWER_STATUS_EX
                {
                        public byte ACLineStatus;
                        public byte BatteryFlag;
                        public byte BatteryLifePercent;
                        public byte Reserved1;
                        public uint BatteryLifeTime;
                        public uint BatteryFullLifeTime;
```

```
                        public byte Reserved2;
                        public byte BackupBatteryFlag;
                        public byte BackupBatteryLifePercent;
                        public byte Reserved3;
                        public uint BackupBatteryLifeTime;
                        public uint BackupBatteryFullLifeTime;
                }


                public class MEMORY_STATUS_EX
                {
                        public UInt32 dwLength;
                        public UInt32 dwMemoryLoad;
                        public UInt32 ullTotalPhys;
                        public UInt32 ullAvailPhys;
                        public UInt32 ullTotalPageFile;
                        public UInt32 ullAvailPageFile;
                        public UInt32 ullTotalVirtual;
                        public UInt32 ullAvailVirtual;
                        public UInt32 ullAvailExtendedVirtual;

                }


                [DllImport("coredll")]
                private static extern uint GetSystemPowerStatusEx(SYSTEM_POWER_STATUS_EX
lpSystemPowerStatus, bool fUpdate);

                [DllImport("coredll")]
                private static extern void GlobalMemoryStatus(MEMORY_STATUS_EX memoryBuffer);



                static void Main()
                {
                        Application.Run(new Form1());
                }



                private void btnStart_Click(object sender, System.EventArgs e)
                {
                        try
                        {
                                // Get sleep time
                                int sleep = Int32.Parse(txtSleep.Text);

                                // Get info on the current date
                                DateTime today = DateTime.Now;
                                string year = today.Year.ToString();
                                string month = today.Month.ToString();
                                string day = today.Day.ToString();
                                string filePath = (year + month + day + "_log.txt");


                                // Write the first two lines in the log file
                                if (chkStore.Checked)
                                {
                                        StreamWriter sw = File.AppendText(filePath);
                                        string firstLine = ("#Information about Current Time,
Battery Power %, AC Line Status, Available Memory (MB) and Storage Card Free Space (MB)");
                                        string secondLine = ("#Time" + "\t\t" + "BPW%" + "\t" +
"AC" + "\t" + "AvM" + "\t" + "SCFS");
                                        sw.WriteLine(firstLine);
                                        sw.WriteLine(secondLine);
                                        sw.Close();
                                }

                                // Get IPAddress and Port Number of remote machine
                                IPAddress ipAddr = IPAddress.Parse(txtIpAddr.Text);
                                IPEndPoint ipep = new IPEndPoint(ipAddr, port);


                                // Create an empty array to write info and send
                                byte[] sendBytes = new byte[0];
```

```
                                SYSTEM_POWER_STATUS_EX spwrs = new SYSTEM_POWER_STATUS_EX();

                                // Start the loop to collect data
                                while ((GetSystemPowerStatusEx(spwrs, false) == 1) && (loop ==
true))
                                {
                                        // Get a timestamp
                                        DateTime tstamp = DateTime.Now;
                                        string time = tstamp.TimeOfDay.ToString();

                                        // Get battery status
                                        int blp = spwrs.BatteryLifePercent;
                                        int acl = spwrs.ACLineStatus;
                                        string battery = blp.ToString();
                                        string acline = acl.ToString();

                                        // Get memory status
                                        MEMORY_STATUS_EX mem = new MEMORY_STATUS_EX();
                                        double availphy;
                                        GlobalMemoryStatus(mem);
                                        availphy = (mem.ullAvailPhys / 1048576); // Convert to
Megabytes
                                        availphy = Math.Round(availphy, 2);
                                        string availmem = availphy.ToString();

                                        // Get storage card status
                                        string[] scardList;
                                        StorageCard.DiskFreeSpace dfs = new
OpenNETCF.IO.StorageCard.DiskFreeSpace();
                                        scardList = StorageCard.GetStorageCardNames();
                                        dfs = StorageCard.GetDiskFreeSpace(scardList[0]);
                                        double freeSpace = (dfs.FreeBytesAvailable / 1048576);
// Convert to Megabytes
                                        freeSpace = Math.Round(freeSpace, 2);
                                        string fs = freeSpace.ToString();

                                        // Write all the collected info in this string
                                        string data = (time + "\t" + battery + "\t" + acline +
"\t" + availmem + "\t" + fs);

                                        // Store data if related checkbox is checked
                                        if (chkStore.Checked)
                                        {
                                                StreamWriter sr = File.AppendText(filePath);
                                                sr.WriteLine(data);
                                                sr.Close();
                                        }

                                        // Send data if related checkbox is checked
                                        if (chkSend.Checked)
                                        {
                                                sendBytes = Encoding.UTF8.GetBytes(data);
                                                client.Send(sendBytes, sendBytes.Length, ipep);
                                        }
                                        // Sleep for "sleep" seconds
                                        Thread.Sleep(sleep);
                                }
                        }
                        catch(Exception)
                        {
                                loop = false;
                                client.Close();
                                Application.Exit();
                        }
                }

                // Enable button START only if there is a sleep time value
                private void txtSleep_TextChanged(object sender, System.EventArgs e)
                {
                        btnStart.Enabled = (txtSleep.Text != "");
                }
        }
}
```

A.5 Status Receiver application Source Code

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Text;

namespace STATUS_RECEIVER
{
	/// <summary>
	/// Summary description for Form1.
	/// </summary>
	public class Form1 : System.Windows.Forms.Form
	{
		//
		private StreamWriter sw = null;
		private string folder = "C:\\Temp";
		private bool loop = true;
		private int port = 11112;
		private UdpClient client = new UdpClient(11112);
		private byte[] recvBytes = new byte[0];
		//
		private System.Windows.Forms.Label lblPort;
		private System.Windows.Forms.TextBox txtPort;
		private System.Windows.Forms.Label lblSavein;
		private System.Windows.Forms.TextBox txtSavein;
		private System.Windows.Forms.Button btnStart;
		/// <summary>
		/// Required designer variable.
		/// </summary>
		private System.ComponentModel.Container components = null;

		public Form1()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after InitializeComponent call
			//
		}

		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if (components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
			this.lblPort = new System.Windows.Forms.Label();
```

```
                        this.txtPort = new System.Windows.Forms.TextBox();
                        this.lblSavein = new System.Windows.Forms.Label();
                        this.txtSavein = new System.Windows.Forms.TextBox();
                        this.btnStart = new System.Windows.Forms.Button();
                        this.SuspendLayout();
                        //
                        // lblPort
                        //
                        this.lblPort.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
                        this.lblPort.Location = new System.Drawing.Point(24, 32);
                        this.lblPort.Name = "lblPort";
                        this.lblPort.Size = new System.Drawing.Size(120, 24);
                        this.lblPort.TabIndex = 0;
                        this.lblPort.Text = "Port Number :";
                        //
                        // txtPort
                        //
                        this.txtPort.Enabled = false;
                        this.txtPort.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
                        this.txtPort.Location = new System.Drawing.Point(144, 28);
                        this.txtPort.Name = "txtPort";
                        this.txtPort.Size = new System.Drawing.Size(64, 26);
                        this.txtPort.TabIndex = 1;
                        this.txtPort.Text = "11112";
                        //
                        // lblSavein
                        //
                        this.lblSavein.Font = new System.Drawing.Font("Microsoft Sans Serif",
12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
                        this.lblSavein.Location = new System.Drawing.Point(24, 76);
                        this.lblSavein.Name = "lblSavein";
                        this.lblSavein.Size = new System.Drawing.Size(72, 24);
                        this.lblSavein.TabIndex = 3;
                        this.lblSavein.Text = "Save in :";
                        //
                        // txtSavein
                        //
                        this.txtSavein.Font = new System.Drawing.Font("Microsoft Sans Serif",
12F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
                        this.txtSavein.Location = new System.Drawing.Point(104, 72);
                        this.txtSavein.Name = "txtSavein";
                        this.txtSavein.Size = new System.Drawing.Size(104, 26);
                        this.txtSavein.TabIndex = 1;
                        this.txtSavein.Text = "C:\\Temp";
                        //
                        // btnStart
                        //
                        this.btnStart.Font = new System.Drawing.Font("Microsoft Sans Serif",
14.25F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
                        this.btnStart.Location = new System.Drawing.Point(24, 112);
                        this.btnStart.Name = "btnStart";
                        this.btnStart.Size = new System.Drawing.Size(184, 40);
                        this.btnStart.TabIndex = 4;
                        this.btnStart.Text = "START";
                        this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
                        //
                        // Form1
                        //
                        this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
                        this.ClientSize = new System.Drawing.Size(232, 174);
                        this.Controls.Add(this.lblSavein);
                        this.Controls.Add(this.btnStart);
                        this.Controls.Add(this.txtPort);
                        this.Controls.Add(this.lblPort);
                        this.Controls.Add(this.txtSavein);
                        this.Name = "Form1";
                        this.Text = "Status Receiver";
                        this.ResumeLayout(false);

                }
                #endregion

                /// <summary>
```

```
                /// The main entry point for the application.
                /// </summary>


                // ***** MAIN *****
                static void Main()
                {
                        Application.Run(new Form1());
                }

                public void ReceiveStatus()
                {
                        while(loop)
                        {
                                // listen for client activity on all network interfaces
                                // on the selected port number
                                IPEndPoint remIpep = new IPEndPoint(IPAddress.Any, port);

                                // Receive data
                                recvBytes = client.Receive(ref remIpep);

                                // Get IPAddress of the remote host
                                string hostIP = remIpep.Address.ToString();

                                // Write the data in the file
                                string filePath = (folder + "\\" + "Source_" + hostIP +
"_onPort_" + txtPort.Text + ".log");
                                sw = File.AppendText(filePath);
                                string data = Encoding.UTF8.GetString(recvBytes, 0,
recvBytes.Length);
                                sw.WriteLine(data);
                                sw.Close();
                        }
                }


                // ***** BUTTON START *****
                private void btnStart_Click(object sender, System.EventArgs e)
                {
                        try
                        {
                                // Check if the selected directory exist.
                                // if not, create new dir
                                folder = txtSavein.Text;
                                if (!Directory.Exists(folder))
                                {
                                        Directory.CreateDirectory(folder);
                                }

                                // Get a timestamp
                                DateTime now = DateTime.Now;
                                string year = now.Year.ToString();
                                string month = now.Month.ToString();
                                string day = now.Day.ToString();

                                // Create a File to explain collected data
                                StreamWriter label = new StreamWriter(folder + "\\" +
"README.txt");
                                string tstamp = ("'Status Receiver' application was started on "
+ year + "-" + month + "-" + day);
                                string description = ("The files
'Source_<IPaddress>_onPort_<portNumber>.log' contain information on the Current Time, Battery
Power %, AC Line Status (1 if connected, otherwise 0), Available Memory (in Megabytes) and Storage
Card Free Space (in Megabytes) respectively, of the device running the 'Status Client'
application. i.e.");
                                string example1 = ("Time" + "\t\t" + "BPW%" + "\t" + "AC" + "\t"
+ "AvM" + "\t" + "SCFS");
                                string example2 = ("hh:mm:ss" + "\t" + "99" + "\t" + "0" + "\t"
+ "53" + "\t" + "16");
                                string example3 = ("...");
                                label.WriteLine(tstamp);
                                label.WriteLine(description);
                                label.WriteLine(example1);
                                label.WriteLine(example2);
                                label.WriteLine(example3);
```

```
                                label.Close();

                                ReceiveStatus();
                        }
                        catch(Exception)
                        {
                                client.Close();
                                Application.Exit();
                        }
                }
        }
}
```

A.6 TCP File Sender application Source Code

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.IO;

namespace TCP_FILE_SENDER
{
        /// <summary>
        /// Summary description for Form1.
        /// </summary>
        public class Form1 : System.Windows.Forms.Form
        {
                private string localDir = "\\";
                private int port = 11115;
                private int bsize = 1024;
                //
                private System.Windows.Forms.TextBox txtIPaddr;
                private System.Windows.Forms.TextBox txtFilename;
                private System.Windows.Forms.Button btnSend;
                private System.Windows.Forms.ColumnHeader columnHeader1;
                private System.Windows.Forms.ColumnHeader columnHeader2;
                private System.Windows.Forms.ListView listView;
                private System.Windows.Forms.TextBox txtLocaldir;
                private System.Windows.Forms.Button btnApply;
                private System.Windows.Forms.Label lblIP;
                private System.Windows.Forms.Label lblFile;
                private System.Windows.Forms.Label lblFolder;
                private System.Windows.Forms.MenuItem menuFile;
                private System.Windows.Forms.MenuItem menuExit;
                private System.Windows.Forms.MainMenu mainMenu1;

                public Form1()
                {
                        //
                        // Required for Windows Form Designer support
                        //
                        InitializeComponent();

                        //
                        // TODO: Add any constructor code after InitializeComponent call
                        //
                }
                /// <summary>
                /// Clean up any resources being used.
                /// </summary>
```

```
		protected override void Dispose( bool disposing )
		{
			base.Dispose( disposing );
		}
		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
			this.mainMenu1 = new System.Windows.Forms.MainMenu();
			this.menuFile = new System.Windows.Forms.MenuItem();
			this.menuExit = new System.Windows.Forms.MenuItem();
			this.txtIPaddr = new System.Windows.Forms.TextBox();
			this.lblIP = new System.Windows.Forms.Label();
			this.txtFilename = new System.Windows.Forms.TextBox();
			this.lblFile = new System.Windows.Forms.Label();
			this.btnSend = new System.Windows.Forms.Button();
			this.listView = new System.Windows.Forms.ListView();
			this.columnHeader1 = new System.Windows.Forms.ColumnHeader();
			this.columnHeader2 = new System.Windows.Forms.ColumnHeader();
			this.txtLocaldir = new System.Windows.Forms.TextBox();
			this.lblFolder = new System.Windows.Forms.Label();
			this.btnApply = new System.Windows.Forms.Button();
			//
			// mainMenu1
			//
			this.mainMenu1.MenuItems.Add(this.menuFile);
			//
			// menuFile
			//
			this.menuFile.MenuItems.Add(this.menuExit);
			this.menuFile.Text = "File";
			//
			// menuExit
			//
			this.menuExit.Text = "Exit";
			this.menuExit.Click += new System.EventHandler(this.menuExit_Click);
			//
			// txtIPaddr
			//
			this.txtIPaddr.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
			this.txtIPaddr.Location = new System.Drawing.Point(120, 8);
			this.txtIPaddr.Size = new System.Drawing.Size(112, 20);
			this.txtIPaddr.Text = "";
			//
			// lblIP
			//
			this.lblIP.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold);
			this.lblIP.Location = new System.Drawing.Point(8, 10);
			this.lblIP.Size = new System.Drawing.Size(96, 24);
			this.lblIP.Text = "IP Address";
			//
			// txtFilename
			//
			this.txtFilename.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
			this.txtFilename.Location = new System.Drawing.Point(56, 40);
			this.txtFilename.Size = new System.Drawing.Size(112, 20);
			this.txtFilename.Text = "";
			//
			// lblFile
			//
			this.lblFile.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
			this.lblFile.Location = new System.Drawing.Point(8, 42);
			this.lblFile.Size = new System.Drawing.Size(40, 24);
			this.lblFile.Text = "File";
			//
			// btnSend
			//
```

```
                        this.btnSend.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
                        this.btnSend.Location = new System.Drawing.Point(176, 40);
                        this.btnSend.Size = new System.Drawing.Size(56, 20);
                        this.btnSend.Text = "SEND";
                        this.btnSend.Click += new System.EventHandler(this.btnSend_Click);
                        //
                        // listView
                        //
                        this.listView.Columns.Add(this.columnHeader1);
                        this.listView.Columns.Add(this.columnHeader2);
                        this.listView.Location = new System.Drawing.Point(8, 88);
                        this.listView.Size = new System.Drawing.Size(224, 168);
                        this.listView.View = System.Windows.Forms.View.Details;
                        //
                        // columnHeader1
                        //
                        this.columnHeader1.Text = "File Name";
                        this.columnHeader1.Width = 132;
                        //
                        // columnHeader2
                        //
                        this.columnHeader2.Text = "Size (Kb)";
                        this.columnHeader2.Width = 90;
                        //
                        // txtLocaldir
                        //
                        this.txtLocaldir.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
                        this.txtLocaldir.Location = new System.Drawing.Point(56, 64);
                        this.txtLocaldir.Size = new System.Drawing.Size(112, 20);
                        this.txtLocaldir.Text = "";
                        //
                        // lblFolder
                        //
                        this.lblFolder.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.lblFolder.Location = new System.Drawing.Point(8, 66);
                        this.lblFolder.Size = new System.Drawing.Size(72, 24);
                        this.lblFolder.Text = "Folder";
                        //
                        // btnApply
                        //
                        this.btnApply.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
                        this.btnApply.Location = new System.Drawing.Point(176, 64);
                        this.btnApply.Size = new System.Drawing.Size(56, 20);
                        this.btnApply.Text = "APPLY";
                        this.btnApply.Click += new System.EventHandler(this.btnApply_Click);
                        //
                        // Form1
                        //
                        this.Controls.Add(this.btnApply);
                        this.Controls.Add(this.listView);
                        this.Controls.Add(this.btnSend);
                        this.Controls.Add(this.lblIP);
                        this.Controls.Add(this.txtIPaddr);
                        this.Controls.Add(this.txtFilename);
                        this.Controls.Add(this.lblFile);
                        this.Controls.Add(this.txtLocaldir);
                        this.Controls.Add(this.lblFolder);
                        this.Menu = this.mainMenu1;
                        this.Text = "TCP File Sender";

                }
                #endregion

                /// <summary>
                /// The main entry point for the application.
                /// </summary>


                // ***** POPULATE *****
                void Populate(string localDir)
                {
```

```
                        try
                        {
                                // Get the files from the selected directory
                                listView.Items.Clear();
                                string[] myFiles = Directory.GetFiles(localDir);

                                for(int i=0; i<myFiles.Length; i++)
                                {
                                        FileInfo fi = new FileInfo(myFiles[i]);
                                        string[] nameAndSize = new string[2];
                                        nameAndSize[0] = fi.Name;
                                        double fSizeKB = (fi.Length / 1024); // Convert to
Kilobytes
                                        fSizeKB = Math.Round(fSizeKB, 2);
                                        nameAndSize[1] = fSizeKB.ToString();
                                        listView.Items.Add(new ListViewItem(nameAndSize));
                                }
                        }
                        catch(Exception e)
                        {
                                MessageBox.Show("Directory not found!", e.Message,
MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
                        }
                }


                // ***** CONNECT *****
                public void Connect()
                {
                        try
                        {
                                // Create a TCP Client and connect
                                TcpClient client = new TcpClient(txtIPaddr.Text, port);

                                // ** Send a message with file name and  size **

                                // Get file name and size
                                FileInfo fi = new FileInfo(txtFilename.Text);
                                string fname = fi.Name;
                                string fsize = fi.Length.ToString();

                                // Create a string to write these information
                                // To divide file name from size is used the null char
                                string finfo = (fname + "\x00" + fsize);

                                // Get the bytes of the string created above
                                byte[] finfob = new byte[bsize];
                                finfob = System.Text.Encoding.ASCII.GetBytes(finfo);

                                // Get a Network Stream for writing
                                NetworkStream stream = client.GetStream();

                                // Send file name and size
                                stream.Write(finfob, 0, finfob.Length);


                                // ** Receive the response **

                                // Buffer to store the response
                                byte[] reply = new byte[bsize];

                                // String to store the response ASCII representation
                                string responseData = null;

                                // Read from the Network Stream to get the response
                                int bytes = stream.Read(reply, 0, reply.Length);
                                responseData = System.Text.Encoding.ASCII.GetString(reply, 0,
bytes);


                                if (responseData == "ACCEPT")
                                {
                                        // Send the file
                                        SendFile(fname, fsize, client, stream);
                                }
```

```
                                if (responseData == "REFUSE")
                                {
                                        // Prompt with a MessageBox
                                        MessageBox.Show("File refused!!", "FILE REFUSED",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);

                                        // Terminate the application
                                        client.Close();
                                        Application.Exit();
                                }
                        }
                        catch(ArgumentNullException e)
                        {
                                MessageBox.Show("Argument null exception: " + e.Message);
                        }
                        catch(SocketException ee)
                        {
                                MessageBox.Show("Socket exception: " + ee.Message);
                        }

                }


                // ***** SEND FILE *****
                void SendFile(String fname, String fsize, TcpClient client, NetworkStream stream)
                {
                        try
                        {
                                long size = int.Parse(fsize);
                                long readBytes = 0;
                                int len;

                                // Open the file to send
                                FileStream fs = new FileStream(txtFilename.Text, FileMode.Open,
FileAccess.Read);

                                // Loop to read all the bytes of the file
                                while(readBytes < size && stream.CanWrite)
                                {
                                        byte[] datatosend = new byte[bsize];

                                        // Read the bytes from the file
                                        len = fs.Read(datatosend, 0, datatosend.Length);

                                        // Write the bytes on the Network Stream
                                        stream.Write(datatosend, 0, len);

                                        // Increase the counter
                                        readBytes = readBytes + len;
                                }
                                fs.Close();


                                // ** Receive the 'TRANSFER COMPLETED' message **

                                // Buffer to store the response
                                byte[] ackb = new byte[bsize];

                                // String to store the response ASCII representation
                                string ack = null;

                                // Read from the Network Stream
                                int ackBytes = stream.Read(ackb, 0, ackb.Length);
                                ack = System.Text.Encoding.ASCII.GetString(ackb, 0, ackBytes);

                                // Prompt with MessageBox
                                DialogResult sendAgain;
                                sendAgain = MessageBox.Show("Send another file?", ack,
MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1);

                                if(sendAgain == DialogResult.No)
                                {
                                        // Send back a "QUIT" message
                                        string qu = "QUIT";
```

```
                                        byte[] quitb = new byte[bsize];
                                        quitb = System.Text.Encoding.ASCII.GetBytes(qu);
                                        stream.Write(quitb, 0, quitb.Length);

                                        // Terminate the application
                                        client.Close();
                                        Application.Exit();
                                }

                                if(sendAgain == DialogResult.Yes)
                                {
                                        // Send back an "ANOTHER" message
                                        string an = "ANOTHER";
                                        byte[] anotherb = new byte[bsize];
                                        anotherb = System.Text.Encoding.ASCII.GetBytes(an);

                                        stream.Write(anotherb, 0, anotherb.Length);
                                }
                        }
                        catch(Exception e)
                        {
                                MessageBox.Show(e.Message);
                                Application.Exit();
                        }
                }


                // ***** MAIN *****
                static void Main()
                {
                        Application.Run(new Form1());
                }


                // ***** BUTTON SEND *****
                private void btnSend_Click(object sender, System.EventArgs e)
                {
                        Connect();
                }


                // ***** BUTTON APPLY *****
                private void btnApply_Click(object sender, System.EventArgs e)
                {
                        if (txtLocaldir.Text == "")
                        {
                                MessageBox.Show("Select a folder!", "EMPTY FIELD",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
                        }
                        else
                        {
                                localDir = txtLocaldir.Text;
                                Populate(localDir);
                        }
                }


                // ***** MENU EXIT *****
                private void menuExit_Click(object sender, System.EventArgs e)
                {
                        Application.Exit();
                }

        }
}
```

A.7 TCP File Receiver application Source Code

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.IO;

namespace TCP_FILE_RECEIVER
{
	/// <summary>
	/// Summary description for Form1.
	/// </summary>
	public class Form1 : System.Windows.Forms.Form
	{
		private int port = 11115;
		private string localDir = "\\My Documents\\MyFiles";
		private string nul = "\x00";
		private int bsize = 1024;
		//
		private System.Windows.Forms.ListView listView;
		private System.Windows.Forms.ColumnHeader columnHeader1;
		private System.Windows.Forms.ColumnHeader columnHeader2;
		private System.Windows.Forms.Button btnStart;
		private System.Windows.Forms.Button btnApply;
		private System.Windows.Forms.TextBox txtLocaldir;
		private System.Windows.Forms.MainMenu mainMenu1;
		private System.Windows.Forms.MenuItem menuItem1;
		private System.Windows.Forms.MenuItem menuItem2;
		private System.Windows.Forms.Label lblSavein;


		public Form1()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after InitializeComponent call
			//
		}
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			base.Dispose( disposing );
		}
		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		private void InitializeComponent()
		{
			this.listView = new System.Windows.Forms.ListView();
			this.columnHeader1 = new System.Windows.Forms.ColumnHeader();
			this.columnHeader2 = new System.Windows.Forms.ColumnHeader();
			this.btnStart = new System.Windows.Forms.Button();
			this.btnApply = new System.Windows.Forms.Button();
			this.txtLocaldir = new System.Windows.Forms.TextBox();
			this.lblSavein = new System.Windows.Forms.Label();
			this.mainMenu1 = new System.Windows.Forms.MainMenu();
			this.menuItem1 = new System.Windows.Forms.MenuItem();
			this.menuItem2 = new System.Windows.Forms.MenuItem();
			//
```

```
                        // listView
                        //
                        this.listView.Columns.Add(this.columnHeader1);
                        this.listView.Columns.Add(this.columnHeader2);
                        this.listView.Location = new System.Drawing.Point(8, 72);
                        this.listView.Size = new System.Drawing.Size(224, 184);
                        this.listView.View = System.Windows.Forms.View.Details;
                        //
                        // columnHeader1
                        //
                        this.columnHeader1.Text = "File Name";
                        this.columnHeader1.Width = 130;
                        //
                        // columnHeader2
                        //
                        this.columnHeader2.Text = "Size (KBytes)";
                        this.columnHeader2.Width = 91;
                        //
                        // btnStart
                        //
                        this.btnStart.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.btnStart.Location = new System.Drawing.Point(8, 8);
                        this.btnStart.Size = new System.Drawing.Size(224, 32);
                        this.btnStart.Text = "START";
                        this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
                        //
                        // btnApply
                        //
                        this.btnApply.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
                        this.btnApply.Location = new System.Drawing.Point(176, 48);
                        this.btnApply.Size = new System.Drawing.Size(56, 20);
                        this.btnApply.Text = "APPLY";
                        this.btnApply.Click += new System.EventHandler(this.btnApply_Click);
                        //
                        // txtLocaldir
                        //
                        this.txtLocaldir.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
                        this.txtLocaldir.Location = new System.Drawing.Point(64, 48);
                        this.txtLocaldir.Size = new System.Drawing.Size(104, 20);
                        this.txtLocaldir.Text = "\\My Documents\\MyFiles";
                        //
                        // lblSavein
                        //
                        this.lblSavein.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.lblSavein.Location = new System.Drawing.Point(8, 50);
                        this.lblSavein.Size = new System.Drawing.Size(56, 14);
                        this.lblSavein.Text = "Save in:";
                        //
                        // mainMenu1
                        //
                        this.mainMenu1.MenuItems.Add(this.menuItem1);
                        //
                        // menuItem1
                        //
                        this.menuItem1.MenuItems.Add(this.menuItem2);
                        this.menuItem1.Text = "File";
                        //
                        // menuItem2
                        //
                        this.menuItem2.Text = "Exit";
                        this.menuItem2.Click += new System.EventHandler(this.menuItem2_Click);
                        //
                        // Form1
                        //
                        this.Controls.Add(this.btnApply);
                        this.Controls.Add(this.txtLocaldir);
                        this.Controls.Add(this.lblSavein);
                        this.Controls.Add(this.btnStart);
                        this.Controls.Add(this.listView);
                        this.Menu = this.mainMenu1;
                        this.Text = "TCP File Receiver";
```

```
		}
		#endregion

		/// <summary>
		/// The main entry point for the application.
		/// </summary>


		// ***** POPULATE *****
		void Populate(string localDir)
		{
			try
			{
				// Clear the listView control
				listView.Items.Clear();

				// Get the files from the selected directory
				string[] myFiles = Directory.GetFiles(localDir);

				for(int i=0; i<myFiles.Length; i++)
				{
					FileInfo fi = new FileInfo(myFiles[i]);
					string[] nameAndSize = new string[2];
					nameAndSize[0] = fi.Name;
					double fSizeKB = (fi.Length / 1024); // Convert to
Kilobytes
					fSizeKB = Math.Round(fSizeKB, 2);
					nameAndSize[1] = fSizeKB.ToString();
					listView.Items.Add(new ListViewItem(nameAndSize));
				}
			}
			catch(Exception e)
			{
				MessageBox.Show(e.Message, "Directory not found!",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
			}
		}


		// ***** START SERVER *****
		public void StartServer()
		{
			try
			{
				// Check if directiry exists
				// If not, create the dir
				if (!Directory.Exists(localDir))
					Directory.CreateDirectory(localDir);

				// Set up a TCP Listener
				IPAddress myIP = IPAddress.Parse("0.0.0.0");
				TcpListener server = new TcpListener(myIP, port);

				// Start listening for client request
				server.Start();

				// Perform a blocking call to accept request
				TcpClient client = server.AcceptTcpClient();

				// Get a Network Stream for reading and writing
				NetworkStream stream = client.GetStream();

				// Buffer for reading data
				byte[] bytes = new byte[bsize];
				string data = null;

				// Read from the stream
				int i = stream.Read(bytes, 0, bytes.Length);
				data = System.Text.Encoding.ASCII.GetString(bytes, 0, i);

				data = data.Substring(0, i);
				char nullchar = nul[0];
				string[] finfo = data.Split(nullchar);
				string fname = finfo[0].ToString();
```

```
                                string fsize = finfo[1].ToString();

                                // Convert bytes to Kilobytes to display
                                double fsizeKB = int.Parse(fsize);
                                fsizeKB = (fsizeKB / 1024);
                                fsizeKB = Math.Round(fsizeKB, 0);
                                string fsKB = fsizeKB.ToString();

                                // Prompt with a MessageBox to choose if accept or refuse the
file
                                DialogResult accept;
                                accept = MessageBox.Show("Accept " + fname + " of size " + fsKB
+ " Kb ?", "ACCEPT FILE?", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1);


                                if(accept == DialogResult.No)
                                {
                                        // Send back a "REFUSE" reply
                                        string reply = "REFUSE";
                                        byte[] replyb = new byte[bsize];
                                        replyb = System.Text.Encoding.ASCII.GetBytes(reply);
                                        stream.Write(replyb, 0, replyb.Length);

                                        // Terminate the application
                                        client.Close();
                                        Application.Exit();
                                }

                                if(accept == DialogResult.Yes)
                                {
                                        // Send back an "ACCEPT" reply
                                        string reply = "ACCEPT";
                                        byte[] replyb = new byte[bsize];
                                        replyb = System.Text.Encoding.ASCII.GetBytes(reply);

                                        stream.Write(replyb, 0, replyb.Length);

                                        // Start receiving the file
                                        ReceiveFile(fname, fsize, client, server, stream);
                                }
                        }
                        catch(Exception es)
                        {
                                MessageBox.Show(es.Message);
                                Application.Exit();
                        }
                }

                // ***** RECEIVE FILE *****
                void ReceiveFile(String fname, String fsize, TcpClient client, TcpListener
server, NetworkStream stream)
                {
                        try
                        {
                                long size = int.Parse(fsize);
                                long readBytes = 0;

                                // Create the file to write the received bytes
                                FileStream fs = new FileStream(txtLocaldir.Text + "\\" + fname,
FileMode.Create, FileAccess.Write);

                                // loop to receive all the data
                                while(readBytes < size)
                                {
                                        byte[] buffer = new byte[bsize];

                                        // Read from the Network Stream
                                        int j = stream.Read(buffer, 0, buffer.Length);

                                        // Write in the File Stream
                                        fs.Write(buffer, 0, j);

                                        // Increase the counter
                                        readBytes = readBytes + j;
```

```
                                }

                                // Close the file
                                fs.Close();

                                // Send a message "TRANSFER COMPLETED"
                                string ok = "TRANSFER COMPLETED";
                                byte[] fileok = new byte[bsize];
                                fileok = System.Text.Encoding.ASCII.GetBytes(ok);
                                stream.Write(fileok, 0, fileok.Length);


                                DialogResult again;
                                again = MessageBox.Show("Continue listening for to receive
another file?", "FILE RECEIVED", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1);

                                if(again == DialogResult.Yes)
                                {
                                        // Buffer for reading data
                                        byte[] by = new byte[bsize];
                                        string str = null;

                                        // Read from the stream
                                        int j = stream.Read(by, 0, by.Length);
                                        str = System.Text.Encoding.ASCII.GetString(by, 0, j);

                                        if(str == "ANOTHER")
                                        {
                                                server.Stop();
                                                client.Close();
                                                StartServer();
                                        }
                                        if(str == "QUIT")
                                        {
                                                server.Stop();
                                                client.Close();
                                                MessageBox.Show("Shutting down application",
"QUIT MESSAGE RECEIVED", MessageBoxButtons.OK, MessageBoxIcon.Exclamation,
MessageBoxDefaultButton.Button1);
                                                Application.Exit();
                                        }

                                        server.Stop();
                                        client.Close();
                                        Application.Exit();
                                }
                                if(again == DialogResult.No)
                                {
                                        server.Stop();
                                        client.Close();
                                        Application.Exit();
                                }
                        }
                        catch(Exception er)
                        {
                                MessageBox.Show(er.Message);
                                Application.Exit();
                        }
                }


                // ***** MAIN *****
                static void Main()
                {
                        Application.Run(new Form1());
                }


                // ***** BUTTON START *****
                private void btnStart_Click(object sender, System.EventArgs e)
                {
                        StartServer();
                }
```

```
                // ***** BUTTON APPLY *****
                private void btnApply_Click(object sender, System.EventArgs e)
                {
                        if (txtLocaldir.Text == "")
                        {
                                MessageBox.Show("Select a folder!", "EMPTY FIELD",
MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
                        }
                        else
                        {
                                localDir = txtLocaldir.Text;
                                Populate(localDir);
                        }
                }

                // ***** MENU EXIT *****
                private void menuItem2_Click(object sender, System.EventArgs e)
                {
                        Application.Exit();
                }
        }
}
```

A.8 UDP Text “Char” application Source Code

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.IO;
using System.Threading;

namespace UDP_TEXT_CHAR
{
        /// <summary>
        /// Summary description for Form1.
        /// </summary>
        public class Form1 : System.Windows.Forms.Form
        {
                private int port = 11117;
                private byte[] bytes = new byte[0];
                private string msg = null;
                private bool loop = true;
                private StreamWriter sw = null;
                private UdpClient client = new UdpClient(11117);
                private System.Windows.Forms.MenuItem menuItem1;
                private System.Windows.Forms.MenuItem menuItem2;
                private System.Windows.Forms.TextBox txtIPaddr;
                private System.Windows.Forms.Label lblIPaddr;
                private System.Windows.Forms.Button btnStart;
                private System.Windows.Forms.TextBox txtOutgoing;
                private System.Windows.Forms.CheckBox chkStore;
                private System.Windows.Forms.Label label1;
                private System.Windows.Forms.Label label2;
                private System.Windows.Forms.Label label3;
                private System.Windows.Forms.Label label4;
                private System.Windows.Forms.Label label5;
                private System.Windows.Forms.Label label6;
                private System.Windows.Forms.Label label7;
                private System.Windows.Forms.Label label8;
```

```
private System.Windows.Forms.Label label9;
private System.Windows.Forms.Label lblOut;
private System.Windows.Forms.Label label10;
private System.Windows.Forms.Label label11;
private System.Windows.Forms.Label label12;
private System.Windows.Forms.Label label13;
private System.Windows.Forms.Label label14;
private System.Windows.Forms.MainMenu mainMenu1;

public Form1()
{
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();

        //
        // TODO: Add any constructor code after InitializeComponent call
        //
}
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
        base.Dispose( disposing );
}
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
        this.mainMenu1 = new System.Windows.Forms.MainMenu();
        this.menuItem1 = new System.Windows.Forms.MenuItem();
        this.menuItem2 = new System.Windows.Forms.MenuItem();
        this.txtIPaddr = new System.Windows.Forms.TextBox();
        this.lblIPaddr = new System.Windows.Forms.Label();
        this.btnStart = new System.Windows.Forms.Button();
        this.txtOutgoing = new System.Windows.Forms.TextBox();
        this.chkStore = new System.Windows.Forms.CheckBox();
        this.label1 = new System.Windows.Forms.Label();
        this.label2 = new System.Windows.Forms.Label();
        this.label3 = new System.Windows.Forms.Label();
        this.label4 = new System.Windows.Forms.Label();
        this.label5 = new System.Windows.Forms.Label();
        this.label6 = new System.Windows.Forms.Label();
        this.label7 = new System.Windows.Forms.Label();
        this.label8 = new System.Windows.Forms.Label();
        this.label9 = new System.Windows.Forms.Label();
        this.lblOut = new System.Windows.Forms.Label();
        this.label10 = new System.Windows.Forms.Label();
        this.label11 = new System.Windows.Forms.Label();
        this.label12 = new System.Windows.Forms.Label();
        this.label13 = new System.Windows.Forms.Label();
        this.label14 = new System.Windows.Forms.Label();
        //
        // mainMenu1
        //
        this.mainMenu1.MenuItems.Add(this.menuItem1);
        //
        // menuItem1
        //
        this.menuItem1.MenuItems.Add(this.menuItem2);
        this.menuItem1.Text = "File";
        //
        // menuItem2
        //
        this.menuItem2.Text = "Exit";
        this.menuItem2.Click += new System.EventHandler(this.menuItem2_Click);
        //
        // txtIPaddr
        //
```

```
                        this.txtIPaddr.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
                        this.txtIPaddr.Location = new System.Drawing.Point(64, 8);
                        this.txtIPaddr.Size = new System.Drawing.Size(104, 20);
                        this.txtIPaddr.Text = "";
                        this.txtIPaddr.TextChanged += new
System.EventHandler(this.txtIPaddr_TextChanged);
                        //
                        // lblIPaddr
                        //
                        this.lblIPaddr.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.lblIPaddr.Location = new System.Drawing.Point(8, 12);
                        this.lblIPaddr.Size = new System.Drawing.Size(56, 24);
                        this.lblIPaddr.Text = "IP Addr";
                        //
                        // btnStart
                        //
                        this.btnStart.Enabled = false;
                        this.btnStart.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
                        this.btnStart.Location = new System.Drawing.Point(176, 8);
                        this.btnStart.Size = new System.Drawing.Size(56, 20);
                        this.btnStart.Text = "START";
                        this.btnStart.Click += new System.EventHandler(this.btnStart_Click);
                        //
                        // txtOutgoing
                        //
                        this.txtOutgoing.Font = new System.Drawing.Font("Microsoft Sans Serif",
12F, System.Drawing.FontStyle.Bold);
                        this.txtOutgoing.ForeColor = System.Drawing.Color.Red;
                        this.txtOutgoing.Location = new System.Drawing.Point(176, 104);
                        this.txtOutgoing.Size = new System.Drawing.Size(56, 26);
                        this.txtOutgoing.Text = "";
                        this.txtOutgoing.WordWrap = false;
                        this.txtOutgoing.TextChanged += new
System.EventHandler(this.txtOutgoing_TextChanged);
                        //
                        // chkStore
                        //
                        this.chkStore.Checked = true;
                        this.chkStore.CheckState = System.Windows.Forms.CheckState.Checked;
                        this.chkStore.Location = new System.Drawing.Point(8, 32);
                        this.chkStore.Size = new System.Drawing.Size(224, 24);
                        this.chkStore.Text = "Store incoming messages in a file";
                        //
                        // label1
                        //
                        this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label1.ForeColor = System.Drawing.Color.Blue;
                        this.label1.Location = new System.Drawing.Point(216, 64);
                        this.label1.Size = new System.Drawing.Size(16, 24);
                        //
                        // label2
                        //
                        this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label2.ForeColor = System.Drawing.Color.Blue;
                        this.label2.Location = new System.Drawing.Point(200, 64);
                        this.label2.Size = new System.Drawing.Size(16, 24);
                        //
                        // label3
                        //
                        this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label3.ForeColor = System.Drawing.Color.Blue;
                        this.label3.Location = new System.Drawing.Point(184, 64);
                        this.label3.Size = new System.Drawing.Size(16, 24);
                        //
                        // label4
                        //
                        this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label4.ForeColor = System.Drawing.Color.Blue;
```

```
                        this.label4.Location = new System.Drawing.Point(168, 64);
                        this.label4.Size = new System.Drawing.Size(16, 24);
                        //
                        // label5
                        //
                        this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label5.ForeColor = System.Drawing.Color.Blue;
                        this.label5.Location = new System.Drawing.Point(152, 64);
                        this.label5.Size = new System.Drawing.Size(16, 24);
                        //
                        // label6
                        //
                        this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label6.ForeColor = System.Drawing.Color.Blue;
                        this.label6.Location = new System.Drawing.Point(136, 64);
                        this.label6.Size = new System.Drawing.Size(16, 24);
                        //
                        // label7
                        //
                        this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label7.ForeColor = System.Drawing.Color.Blue;
                        this.label7.Location = new System.Drawing.Point(120, 64);
                        this.label7.Size = new System.Drawing.Size(16, 24);
                        //
                        // label8
                        //
                        this.label8.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label8.ForeColor = System.Drawing.Color.Blue;
                        this.label8.Location = new System.Drawing.Point(104, 64);
                        this.label8.Size = new System.Drawing.Size(16, 24);
                        //
                        // label9
                        //
                        this.label9.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label9.ForeColor = System.Drawing.Color.Blue;
                        this.label9.Location = new System.Drawing.Point(88, 64);
                        this.label9.Size = new System.Drawing.Size(16, 24);
                        //
                        // lblOut
                        //
                        this.lblOut.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold);
                        this.lblOut.Location = new System.Drawing.Point(8, 108);
                        this.lblOut.Size = new System.Drawing.Size(168, 24);
                        this.lblOut.Text = "Type here outgoing text :";
                        //
                        // label10
                        //
                        this.label10.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label10.ForeColor = System.Drawing.Color.Blue;
                        this.label10.Location = new System.Drawing.Point(72, 64);
                        this.label10.Size = new System.Drawing.Size(16, 24);
                        //
                        // label11
                        //
                        this.label11.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label11.ForeColor = System.Drawing.Color.Blue;
                        this.label11.Location = new System.Drawing.Point(56, 64);
                        this.label11.Size = new System.Drawing.Size(16, 24);
                        //
                        // label12
                        //
                        this.label12.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label12.ForeColor = System.Drawing.Color.Blue;
                        this.label12.Location = new System.Drawing.Point(40, 64);
                        this.label12.Size = new System.Drawing.Size(16, 24);
                        //
```

```
                        // label13
                        //
                        this.label13.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label13.ForeColor = System.Drawing.Color.Blue;
                        this.label13.Location = new System.Drawing.Point(24, 64);
                        this.label13.Size = new System.Drawing.Size(16, 24);
                        //
                        // label14
                        //
                        this.label14.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.label14.ForeColor = System.Drawing.Color.Blue;
                        this.label14.Location = new System.Drawing.Point(8, 64);
                        this.label14.Size = new System.Drawing.Size(16, 24);
                        //
                        // Form1
                        //
                        this.Controls.Add(this.lblOut);
                        this.Controls.Add(this.label1);
                        this.Controls.Add(this.chkStore);
                        this.Controls.Add(this.txtOutgoing);
                        this.Controls.Add(this.btnStart);
                        this.Controls.Add(this.lblIPaddr);
                        this.Controls.Add(this.txtIPaddr);
                        this.Controls.Add(this.label2);
                        this.Controls.Add(this.label3);
                        this.Controls.Add(this.label4);
                        this.Controls.Add(this.label5);
                        this.Controls.Add(this.label6);
                        this.Controls.Add(this.label7);
                        this.Controls.Add(this.label8);
                        this.Controls.Add(this.label9);
                        this.Controls.Add(this.label10);
                        this.Controls.Add(this.label11);
                        this.Controls.Add(this.label12);
                        this.Controls.Add(this.label13);
                        this.Controls.Add(this.label14);
                        this.Menu = this.mainMenu1;
                        this.Text = "UDP Text \"char\"";

                }
                #endregion

                /// <summary>
                /// The main entry point for the application.
                /// </summary>


                // ***** RECEIVE *****
                void Receive()
                {
                        try
                        {
                                // Set up IPEndPoint
                                IPAddress ipAddr = IPAddress.Parse(txtIPaddr.Text);
                                IPEndPoint ipep = new IPEndPoint(ipAddr, port);

                                // Create the file to store incoming messages
                                if(chkStore.Checked)
                                {
                                        sw = File.AppendText("\\My Documents\\" + txtIPaddr.Text
+ "-UDP_TEXT_CHAR_log.txt");
                                        sw.WriteLine(" ");
                                }

                                while(loop)
                                {
                                        // Receive string
                                        bytes = client.Receive(ref ipep);
                                        msg = Encoding.UTF8.GetString(bytes, 0, bytes.Length);

                                        // Show the string
                                        label14.Text = label13.Text;
                                        label13.Text = label12.Text;
```

```
                        label12.Text = label11.Text;
                        label11.Text = label10.Text;
                        label10.Text = label9.Text;
                        label9.Text = label8.Text;
                        label8.Text = label7.Text;
                        label7.Text = label6.Text;
                        label6.Text = label5.Text;
                        label5.Text = label4.Text;
                        label4.Text = label3.Text;
                        label3.Text = label2.Text;
                        label2.Text = label1.Text;
                        label1.Text = msg;

                        // Store the string
                        if(chkStore.Checked)
                        {
                                sw.Write(msg);
                        }
                }
        }
        catch(Exception er)
        {
                MessageBox.Show(er.Message, "RECEIVE EXCEPTION");
        }
}


// ***** SEND *****
void Send()
{
        try
        {
                // Encode and send the message
                msg = txtOutgoing.Text;
                bytes = Encoding.UTF8.GetBytes(msg);

                string remIP = txtIPaddr.Text;
                client.Send(bytes, bytes.Length, remIP, port);
                txtOutgoing.Text = "";
        }
        catch(Exception es)
        {
                MessageBox.Show(es.Message, "SEND EXCEPTION");
        }
}


// ***** MAIN *****
static void Main()
{
        Application.Run(new Form1());
}


// ***** BUTTON START *****
private void btnStart_Click(object sender, System.EventArgs e)
{
        Thread t1 = new Thread(new ThreadStart(Receive));
        t1.Start();
}


// ***** MENU EXIT *****
private void menuItem2_Click(object sender, System.EventArgs e)
{
        loop = false;
        sw.Close();
        client.Close();
        Application.Exit();
}


// ***** Enable Button Start *****
private void txtIPaddr_TextChanged(object sender, System.EventArgs e)
{
```

```
                btnStart.Enabled = (txtIPaddr.Text != "");
        }

        // ***** TYPE TEXT *****
        private void txtOutgoing_TextChanged(object sender, System.EventArgs e)
        {
                Thread t2 = new Thread(new ThreadStart(Send));
                t2.Start();
        }

    }
}
```

A.9 RTP Text application Source Code

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;

namespace RTP_TEXT
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        // ******************************************************
        //
        // In this application the PAYLOAD TYPE is set to
        // 98 for T140 and 100 for RED for simplicity of
        // implementation, even if the RFC 4103 specifies it is
        // a dynamic payload type, and it should be setted using
        // a SDP procedure with "a=rtpmap"
        //
        // for example:
        //
        // m=text 11000 RTP/AVP 98 100
        // a=rtpmap:98 t140/1000
        // a=rtpmap:100 red/1000
        // a=fmtp:100 98/98
        //
        // for more information see RFC 4102, 4103
        //
        // ******************************************************
        //
        // ** RTP HEADER **
        //
        private BitArray VER = new BitArray(2); // Version
        private BitArray P = new BitArray(1); // Padding
        private BitArray X = new BitArray(1); // Extension
        private BitArray CC = new BitArray(4); // CSRC Count
        private BitArray M = new BitArray(1); // Marker
        private BitArray PT = new BitArray(7); // Payload Type
        private byte[] SN = new byte[2]; // Sequence Number
        private byte[] TS = new byte[4]; // Timestamp
        private byte[] SSRC = new byte[4]; // SSRC Identifier
        private Random ssrc = new Random(); // Random SSRC number
        //
        // ** RTP PAYLOAD **
```

```
//
// Redundant 2
private BitArray R2F = new BitArray(1); // Red 2 F
private BitArray R2PT = new BitArray(7); // Red 2 Payload Type
private BitArray R2TSOS = new BitArray(14); // Red 2 Timestamp Offset
private BitArray R2BLEN = new BitArray(10); // Red 2 Block Length
private byte[] R2 = new byte[0]; // Red 2 Block
// Redundant 1
private BitArray R1F = new BitArray(1); // Red 1 F
private BitArray R1PT = new BitArray(7); // Red 1 Payload Type
private BitArray R1TSOS = new BitArray(14); // Red 1 Timestamp Offset
private BitArray R1BLEN = new BitArray(10); // Red 1 Block Length
private byte[] R1 = new byte[0]; // Red 1 Block
// Primary
private BitArray PRF = new BitArray(1); // Primary F
private BitArray PRPT = new BitArray(7); // Primary Payload Type
private byte[] PR = new byte[0]; // Primary Block
//
private BitArray T140 = new BitArray(7);
private BitArray RED = new BitArray(7);
private byte[] msg = new byte[0];
private int port = 11118;
private ushort one = 1;
private ushort snu;
private string pr = null;
private int ts;
private int ts1;
private int ts2;
private int r1tsos;
private int r2tsos;
private int r1blen;
private int r2blen;
private UdpClient client = new UdpClient(11118);
private IPEndPoint ipep = null;
private bool loop = true;
private byte[] incom = new byte[0];
private string inMsg = null;
//
private System.Windows.Forms.TextBox textBox1;
private System.Windows.Forms.MenuItem menuItem1;
private System.Windows.Forms.MenuItem menuItem3;
private System.Windows.Forms.TextBox txtIPaddr;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Button button1;
private System.Windows.Forms.Button button2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label labelIdle;
private System.Windows.Forms.MainMenu mainMenu1;

public Form1()
{
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();

        //
        // TODO: Add any constructor code after InitializeComponent call
        //
}
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
        base.Dispose( disposing );
}
#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
```

```
                /// the contents of this method with the code editor.
                /// </summary>
                private void InitializeComponent()
                {
                        this.mainMenu1 = new System.Windows.Forms.MainMenu();
                        this.menuItem1 = new System.Windows.Forms.MenuItem();
                        this.menuItem3 = new System.Windows.Forms.MenuItem();
                        this.textBox1 = new System.Windows.Forms.TextBox();
                        this.txtIPaddr = new System.Windows.Forms.TextBox();
                        this.label1 = new System.Windows.Forms.Label();
                        this.label2 = new System.Windows.Forms.Label();
                        this.button1 = new System.Windows.Forms.Button();
                        this.button2 = new System.Windows.Forms.Button();
                        this.label3 = new System.Windows.Forms.Label();
                        this.label4 = new System.Windows.Forms.Label();
                        this.label5 = new System.Windows.Forms.Label();
                        this.label6 = new System.Windows.Forms.Label();
                        this.label7 = new System.Windows.Forms.Label();
                        this.labelIdle = new System.Windows.Forms.Label();
                        //
                        // mainMenu1
                        //
                        this.mainMenu1.MenuItems.Add(this.menuItem1);
                        //
                        // menuItem1
                        //
                        this.menuItem1.MenuItems.Add(this.menuItem3);
                        this.menuItem1.Text = "File";
                        //
                        // menuItem3
                        //
                        this.menuItem3.Text = "Exit";
                        this.menuItem3.Click += new System.EventHandler(this.menuItem3_Click);
                        //
                        // textBox1
                        //
                        this.textBox1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
System.Drawing.FontStyle.Bold);
                        this.textBox1.ForeColor = System.Drawing.Color.Red;
                        this.textBox1.Location = new System.Drawing.Point(176, 96);
                        this.textBox1.Multiline = true;
                        this.textBox1.Size = new System.Drawing.Size(56, 24);
                        this.textBox1.Text = "";
                        //
                        // txtIPaddr
                        //
                        this.txtIPaddr.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.txtIPaddr.Location = new System.Drawing.Point(64, 16);
                        this.txtIPaddr.Size = new System.Drawing.Size(128, 22);
                        this.txtIPaddr.Text = "";
                        this.txtIPaddr.TextChanged += new
System.EventHandler(this.txtIPaddr_TextChanged);
                        //
                        // label1
                        //
                        this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold);
                        this.label1.Location = new System.Drawing.Point(8, 20);
                        this.label1.Size = new System.Drawing.Size(56, 24);
                        this.label1.Text = "IP addr";
                        //
                        // label2
                        //
                        this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold);
                        this.label2.Location = new System.Drawing.Point(88, 92);
                        this.label2.Size = new System.Drawing.Size(88, 32);
                        this.label2.Text = "Type here text to send";
                        //
                        // button1
                        //
                        this.button1.Font = new System.Drawing.Font("Microsoft Sans Serif", 6F,
System.Drawing.FontStyle.Regular);
                        this.button1.Location = new System.Drawing.Point(192, 16);
```

```
                        this.button1.Size = new System.Drawing.Size(40, 22);
                        this.button1.Text = "APPLY";
                        this.button1.Click += new System.EventHandler(this.button1_Click);
                        //
                        // button2
                        //
                        this.button2.Font = new System.Drawing.Font("Microsoft Sans Serif",
9.75F, System.Drawing.FontStyle.Bold);
                        this.button2.Location = new System.Drawing.Point(8, 96);
                        this.button2.Size = new System.Drawing.Size(56, 24);
                        this.button2.Text = "Start";
                        this.button2.Click += new System.EventHandler(this.button2_Click);
                        //
                        // label3
                        //
                        this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold);
                        this.label3.ForeColor = System.Drawing.Color.Blue;
                        this.label3.Location = new System.Drawing.Point(184, 56);
                        this.label3.Size = new System.Drawing.Size(40, 24);
                        this.label3.TextAlign = System.Drawing.ContentAlignment.TopCenter;
                        //
                        // label4
                        //
                        this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold);
                        this.label4.ForeColor = System.Drawing.Color.Blue;
                        this.label4.Location = new System.Drawing.Point(144, 56);
                        this.label4.Size = new System.Drawing.Size(40, 24);
                        this.label4.TextAlign = System.Drawing.ContentAlignment.TopCenter;
                        //
                        // label5
                        //
                        this.label5.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold);
                        this.label5.ForeColor = System.Drawing.Color.Blue;
                        this.label5.Location = new System.Drawing.Point(104, 56);
                        this.label5.Size = new System.Drawing.Size(40, 24);
                        this.label5.TextAlign = System.Drawing.ContentAlignment.TopCenter;
                        //
                        // label6
                        //
                        this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold);
                        this.label6.ForeColor = System.Drawing.Color.Blue;
                        this.label6.Location = new System.Drawing.Point(64, 56);
                        this.label6.Size = new System.Drawing.Size(40, 24);
                        this.label6.TextAlign = System.Drawing.ContentAlignment.TopCenter;
                        //
                        // label7
                        //
                        this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Bold);
                        this.label7.ForeColor = System.Drawing.Color.Blue;
                        this.label7.Location = new System.Drawing.Point(24, 56);
                        this.label7.Size = new System.Drawing.Size(40, 24);
                        this.label7.TextAlign = System.Drawing.ContentAlignment.TopCenter;
                        //
                        // labelIdle
                        //
                        this.labelIdle.Font = new System.Drawing.Font("Microsoft Sans Serif",
8.25F, System.Drawing.FontStyle.Bold);
                        this.labelIdle.ForeColor = System.Drawing.Color.Red;
                        this.labelIdle.Location = new System.Drawing.Point(8, 136);
                        this.labelIdle.Size = new System.Drawing.Size(224, 24);
                        this.labelIdle.TextAlign = System.Drawing.ContentAlignment.TopCenter;
                        //
                        // Form1
                        //
                        this.Controls.Add(this.labelIdle);
                        this.Controls.Add(this.label3);
                        this.Controls.Add(this.button2);
                        this.Controls.Add(this.button1);
                        this.Controls.Add(this.label2);
                        this.Controls.Add(this.label1);
```

```
            this.Controls.Add(this.txtIPaddr);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.label6);
            this.Controls.Add(this.label7);
            this.Menu = this.mainMenu1;
            this.Text = "RTP Text";

    }
    #endregion

    /// <summary>
    /// The main entry point for the application.
    /// </summary>


    // ***** CREATE MSG *****
    void CreateMsg()
    {
            begin:
            // Get the text
            pr = textBox1.Text;

            // Encodes the text and put in Primary block
            PR = Encoding.UTF8.GetBytes(pr);

            // Get a timestamp value (in milliseconds)
            ts = Timestamp();
            object tso = ts;
            TS = (byte[]) tso;

            // Clear textBox1
            textBox1.Text = "";

            // Set M-bit
            M.Set(0,true);

            // Set Payload Type to T140
            PT = T140;

            // Get a random sequence number
            Random sn = new Random();
            object sno = sn;
            snu = (ushort)sno;
            SN = (byte[]) sno;

            // Write all the field in the message to send
            VER.CopyTo(msg, 0);
            P.CopyTo(msg, 2);
            X.CopyTo(msg, 3);
            CC.CopyTo(msg, 4);
            M.CopyTo(msg, 8);
            PT.CopyTo(msg, 9);
            SN.CopyTo(msg, 16);
            TS.CopyTo(msg, 32);
            SSRC.CopyTo(msg, 64);
            PR.CopyTo(msg, 96);

            Send(msg);


            // Write Redundance 1
            R1 = PR;

            // Old 1 timestamp
            ts1 = ts;

            // Buffer for 300 ms
            Thread.Sleep(300);

            // Get the new text and encodes in Primary block
            pr = textBox1.Text;
            PR = Encoding.UTF8.GetBytes(pr);
```

```
// Get a timestamp value (in milliseconds)
ts = Timestamp();
object tso1 = ts;
TS = (byte[]) tso1;

// clear textBox1
textBox1.Text = "";

// Unset M-bit
M.Set(0,false);

// Set Payload Type to RED
PT = RED;

// Sequence Number of primary = SN + 1
snu = (ushort)(snu + one);
object sno1 = snu;
SN = (byte[]) sno1;

R1F.Set(0,true);

// Set R1 Payload Type to T140
R1PT = T140;

// timestamp offset of R1
r1tsos = ts - ts1;
object r1tsoso = r1tsos;
R1TSOS = (BitArray) r1tsoso;

// R1 block length
r1blen = R1.Length;
object r1bleno = r1blen;
R1BLEN = (BitArray) r1bleno;

PRF.Set(0,false);

// Set Primary Payload Type to T140
PRPT = T140;

// Write all the field in the message to send
VER.CopyTo(msg, 0);
P.CopyTo(msg, 2);
X.CopyTo(msg, 3);
CC.CopyTo(msg, 4);
M.CopyTo(msg, 8);
PT.CopyTo(msg, 9);
SN.CopyTo(msg, 16);
TS.CopyTo(msg, 32);
SSRC.CopyTo(msg, 64);
R1F.CopyTo(msg, 96);
R1PT.CopyTo(msg, 97);
R1TSOS.CopyTo(msg, 104);
R1BLEN.CopyTo(msg, 118);
PRF.CopyTo(msg, 119);
PRPT.CopyTo(msg, 120);
R1.CopyTo(msg, 127);
PR.CopyTo(msg, 127+R1.Length);

Send(msg);

// Buffer for 300 ms
Thread.Sleep(300);

// Here start the loop
do
{
        // Write Redundance 1 and 2
        R2 = R1;
        R1 = PR;

        // Old 1 and 2 timestamp
        ts2 = ts1;
        ts1 = ts;
```

```
// Get the new text and encodes in Primary block
pr = textBox1.Text;
PR = Encoding.UTF8.GetBytes(pr);

// Get a timestamp value (in milliseconds)
ts = Timestamp();
object tso2 = ts;
TS = (byte[]) tso2;

// clear textBox1
textBox1.Text = "";

// Sequence Number of primary = SN + 1
snu = (ushort)(snu + one);
object sno2 = snu;
SN = (byte[]) sno2;

R2F.Set(0,true);

// Set R2 Payload Type to T140
R2PT = T140;

// timestamp offset of R2
r2tsos = ts - ts2;
object r2tsoso = r2tsos;
R2TSOS = (BitArray) r2tsoso;

// R2 block length
r2blen = R2.Length;
object r2bleno = r2blen;
R2BLEN = (BitArray) r1bleno;

// Set R1 Payload Type to T140
R1PT = T140;

// timestamp offset of R1
r1tsos = ts - ts1;
object r1tsoso1 = r1tsos;
R1TSOS = (BitArray) r1tsoso1;

// R1 block length
r1blen = R1.Length;
object r1bleno1 = r1blen;
R1BLEN = (BitArray) r1bleno1;

PRF.Set(0,false);

// Set Primary Payload Type to T140
PRPT = T140;

// Write all the field in the message to send
VER.CopyTo(msg, 0);
P.CopyTo(msg, 2);
X.CopyTo(msg, 3);
CC.CopyTo(msg, 4);
M.CopyTo(msg, 8);
PT.CopyTo(msg, 9);
SN.CopyTo(msg, 16);
TS.CopyTo(msg, 32);
SSRC.CopyTo(msg, 64);
R2F.CopyTo(msg, 96);
R2PT.CopyTo(msg, 97);
R2TSOS.CopyTo(msg, 104);
R2BLEN.CopyTo(msg, 118);
R1F.CopyTo(msg, 128);
R1PT.CopyTo(msg, 129);
R1TSOS.CopyTo(msg, 136);
R1BLEN.CopyTo(msg, 150);
PRF.CopyTo(msg, 151);
PRPT.CopyTo(msg, 152);
R2.CopyTo(msg, 159);
R1.CopyTo(msg, 159+R2.Length);
PR.CopyTo(msg, 159+R2.Length+R1.Length);
```

```
                Send(msg);

                // Buffer for 300 ms
                Thread.Sleep(300);
        }
        while((R2.Length == 0) && (R1.Length == 0) && (PR.Length == 0));

        do
        {
                labelIdle.Text = ("IDLE MODE");
        }
                while(textBox1.Text == "");

        goto begin;
}


// ***** SEND MESSAGE *****
void Send(byte[] msg)
{
        try
        {
                client.Send(msg, msg.Length, ipep);
        }
        catch(Exception e)
        {
                MessageBox.Show(e.Message);
        }
}


// ***** RECEIVE *****
void Receive()
{
        try
        {
                while(loop)
                {
                        // Receive string
                        incom = client.Receive(ref ipep);
                        inMsg = Encoding.UTF8.GetString(incom, 0, incom.Length);

                        // Show the string
                        label7.Text = label6.Text;
                        label6.Text = label5.Text;
                        label5.Text = label4.Text;
                        label4.Text = label3.Text;
                        label3.Text = inMsg;
                }
        }
        catch(Exception er)
        {
                MessageBox.Show(er.Message, "RECEIVE EXCEPTION");
        }
}



// ***** TIMESTAMP *****
int Timestamp()
{
        // Get a timestamp value (in milliseconds)
        DateTime now = DateTime.Now;
        int hour = now.Hour;
        int min = now.Minute;
        int sec = now.Second;
        int ms = now.Millisecond;
        int tstamp = (hour*3600000) + (min*60000) + (sec*1000) + ms;
        return(tstamp);
}


// ***** MAIN *****
static void Main()
{
```

```
                Application.Run(new Form1());
            }


            // ***** MENU EXIT *****
            private void menuItem3_Click(object sender, System.EventArgs e)
            {
                loop = false;
                client.Close();
                Application.Exit();
            }


            // ***** BUTTON APPLY *****
            private void button1_Click(object sender, System.EventArgs e)
            {
                // Set RTP Header
                VER.Set(0,true);
                VER.Set(1,false);
                P.Set(0,false);
                X.Set(0,false);
                CC.SetAll(false);
                T140.Set(0, true);
                T140.Set(1, true);
                T140.Set(2, false);
                T140.Set(3, false);
                T140.Set(4, false);
                T140.Set(5, true);
                T140.Set(6, false);
                RED.Set(0, true);
                RED.Set(1, true);
                RED.Set(2, false);
                RED.Set(3, false);
                RED.Set(4, true);
                RED.Set(5, false);
                RED.Set(6, false);
                ssrc.NextBytes(SSRC);

                ipep = new IPEndPoint(IPAddress.Parse(txtIPaddr.Text), port);
            }


            // ***** Enable button Apply *****
            private void txtIPaddr_TextChanged(object sender, System.EventArgs e)
            {
                button1.Enabled = (txtIPaddr.Text != "");
            }


            // ***** BUTTON START *****
            private void button2_Click(object sender, System.EventArgs e)
            {
                Thread t1 = new Thread(new ThreadStart(CreateMsg));
                t1.Start();

                Thread t2 = new Thread(new ThreadStart(Receive));
                t2.Start();
            }
        }
}
```

IMIT/LCN 2005-18

www.kth.se