# CHAPTER 5

# IMPLEMENTATION OF LOCAL AND REMOTE DATA BACKUP SYSTEM

# CHAPTER 5

# IMPLEMENTATION OF LOCAL AND REMOTE DATA BACKUP SYSTEM

## 5.1 INTRODUCTION

Based on the literature review, case study, observation and design, Local and Remote Data Backup System was developed. This chapter will discuss each module of local and remote data backup system and its requirements. Algorithms, function processes and flow charts are included in each module.

## 5.2 REQUIREMENTS OF SYSTEM

The system requirements for Local and Remote Data Backup System are illustrated in Table 5.1.

Table 5.1 Recommended Requirement for Local and Remote Backup System

| Element | Requirement |
|---|---|
| Processor Speed | Pentium II |
| Free Disk Space | 5MB |
| Memory | 64 MB RAM |
| Operating System | Windows 98 |
| Networking | LAN and Internet Access Ready |
| Code | Microsoft Visual Basic 6.0 |

## 5.3 LOCAL AND REMOTE DATA BACKUP SYSTEM

In respect to the synthesis of Local and Remote Data Backup System in chapter 2, the following modules have been determined and developed:

i) Local data backup module
ii) Remote data backup and restore module
iii) Scheduled and Incremental backup module
iv) Backup log files module

### 5.3.1 LOCAL DATA BACKUP MODULE

This module will backup the selected files or directories in a local workstation to any storage devices that are attached to the workstation or reside in the LAN. Firstly, a user must determine the files or directories to be backed up and the destination where the file is stored. The destination may be the optical media storage attached on the local workstation or the backup server in the local area network. Both the directory listings in local workstation and network neighborhood are provided. These are as illustrated in Figure 5.1.

After selecting the source file and its destination, there are three options provided to perform the local data backup module:

i) Scheduled backup
ii) Incremental backup
iii) Manual backup

Both the scheduled backup and incremental backup are discussed later in this chapter. A manual backup is performed whenever the user activates it. The **FileCopy** statement in Visual Basic can be used to perform the operation.

Statement: **FileCopy** [47]

Syntax: **FileCopy** *source, destination*

| | |
|---|---|
| *source* | String expression that specifies the name of the file to be copied. The *source* may include directory or folder, and drive. |
| *destination* | String expression that specifies the target file name. The *destination* may include directory or folder, and drive. |

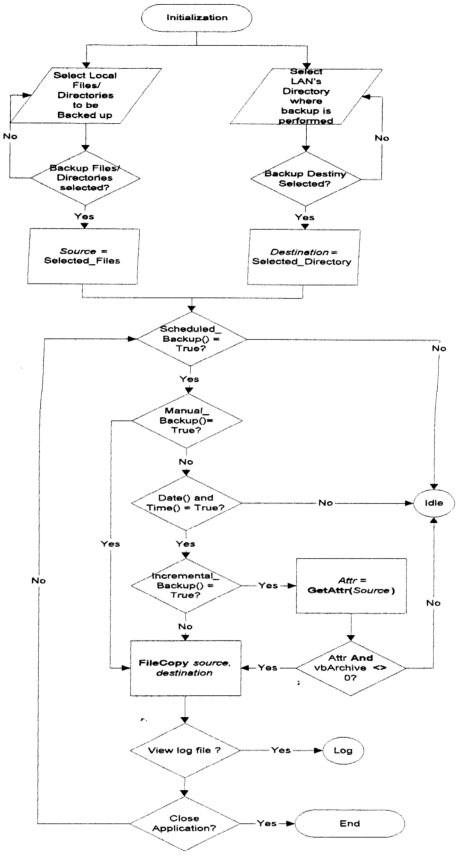


Figure 5.1 Local Data Backup Module Flow Chart

### 5.3.2 REMOTE DATA BACKUP AND RESTORE MODULE

This module performs a backup and restores file operation to or from a remote FTP server. Three sub-modules have been defined, they are:

i) FTP connection setup module
ii) Remote data backup module
iii) Remote data restore module

This remote data backup and restore strategy makes use of the File Transfer Protocol. In general, this module acts as an FTP client and allows communication to the FTP server. Figure 5.2 illustrates FTP transactions between client and server. The development of remote data backup and restore module is based on this algorithm.

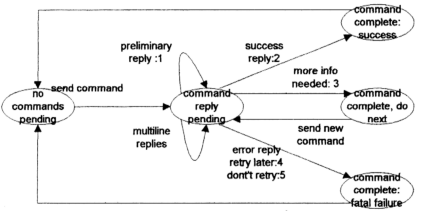


Figure 5.2 State Diagram for FTP Commands and Replies Transaction

### 5.3.2.1 FTP CONNECTION SETUP SUB-MODULE

This sub-module is responsible for initiating control and data connection to the FTP server. Users have to enter the FTP server host name, user name, password and port number in order to log on to the private FTP server. However, in the case of public servers, the common practice is to log in as "anonymous," and send an e-mail address as the password. Port number should set to 21, as it is default port number for FTP. The flow chart of FTP connection setup is shown in Figure 5.3.

In this sub-module, the *Microsoft Internet Transfer Control*, which provides an implementation for two of the most widely used protocols on the Internet, Hyper Text Transfer Protocol (HTTP) and File Transfer Protocol (FTP) is used. With the FTP protocol, files can be downloaded and uploaded with authentication control. Besides, it also supports common FTP commands, such as *CD, PUT* and *GET*, through the *Execute* method. Followings list the properties that are implemented in coding of this module.

Table 5.2 Implemented Internet Transfer Control Properties[48]

| **Property** | **Syntax** | **Implemented** |
|---|---|---|
| Protocol | *Object*.**Protocol** = *integer* | FrmMain.Inet1.Protocol = icFTP (Default) |
| Remote Host | *Object*.**RemoteHost** = *string* | FrmMain.Inet1.RemoteHost = Host (User Defined) |
| UserName | *Object*.**UserName**[= *name*] | FrmMain.Inet1.UserName = User (User Defined) |
| Password | *Object*.**Password** = *string* | FrmMain.Inet1.Password = Passwd (User Defined) |
| RemotePort | *Object*.**RemotePort** = *port* | FrmMain.Inet1.RemotePort = Port (User Defined) |
| RequestTimeout | *Object*.**RequestTimeout** = *time* | FrmMain.Inet1.RequestTimeout = 45 |

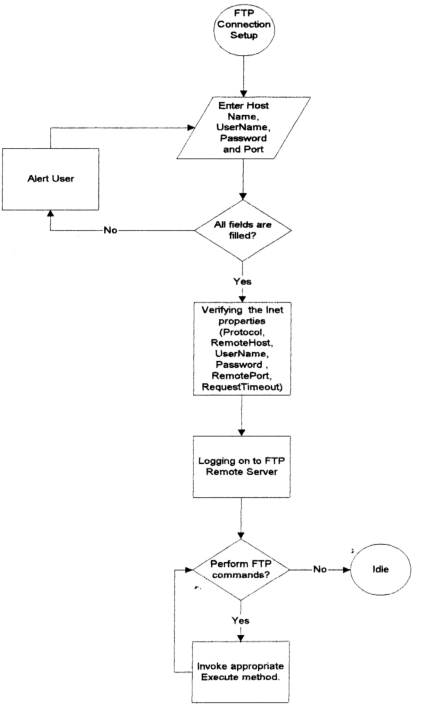


Figure 5.3 FTP Connection Setup Flow Chart

### 5.3.2.2 REMOTE DATA BACKUP SUB-MODULE

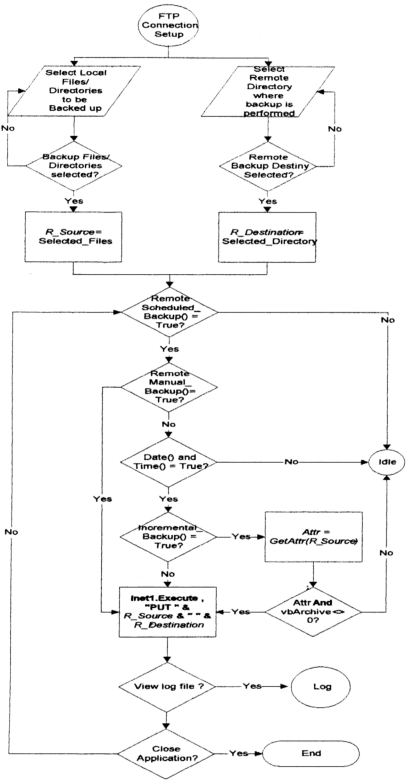


Figure 5.4 Remote Data Backup Sub-Module Flow Chart

As illustrated in Figure 5.4, remote data backup sub-module will perform remote backup for the selected files or directories in the local workstation to a remote server over the Internet. After a successful connection setup to the FTP server, the **Execute** methods of the Internet Transfer Control is invoked to perform the FTP commands.

Method: **Execute** [48]

Syntax: *object.***Execute** *url, operation, data, requestHeaders*

| | |
|---|---|
| *url* | String that specifies the URL to which the control should connect. |
| *operation* | String that specifies the type of operation to be executed. |
| *data* | String that specifies the data for operations |
| *requestHeader* | String that specifies additional headers to be sent from the remote server. |

The first operation invoked in *Execute* method is *DIR* that returns a full directory of the current working directory in remote server. This allows users to select the remote server directory where the local backup files are stored in. Similar to the local data backup module, three kind of backup options are also provided in this module. There are:

i) Scheduled backup
ii) Incremental backup
iii) Manual backup

Both of the scheduled backup and incremental backup are discussed later in this chapter. Manual remote backup will perform whenever the user activates it. The *Put* operation can be used to fulfil this requirement.

Operation: **Put** [48]

Syntax: **Put** *file1 file2*

This will copy a local file specified in *file1* to the remote host specified in *file2*.

### 5.3.2.3 REMOTE DATA RESTORE SUB-MODULE

File restoration from the remote server to the local workstation will be performed. As in the case of remote data backup sub-module, after a successful connection to the FTP server, the *Execute* methods of Internet Transfer Control can be invoked to perform the FTP commands.

The first operation invoked in *Execute* method is *DIR* that returns a full directory of the current working directory in remote server. This allows user to select the remote server file and restore it to the local directory.

In order to browse the directory of remote server, the *CD* operation is used to change directory. This operation is invoked whenever the user clicks on the remote directories to browse files in it.

Operation: **CD** [48]

Syntax: **CD** *file1*

Changes to the directory specified in *file1*.

Unlike the remote backup, remote restore utilizes the *Get* operation whenever the user activates the remote restoration process.

Operation: **Get** [48]

Syntax: **Get** *file1 file2*

This will retrieve the remote file specified in *file1*, and creates a new local file specified in *file2*.

When all of the operations in the remote data backup and restore module have been performed, invoking the Close operation can close the connection from the remote server. The flow chart of this sub-module is depicted in Figure 5.5.

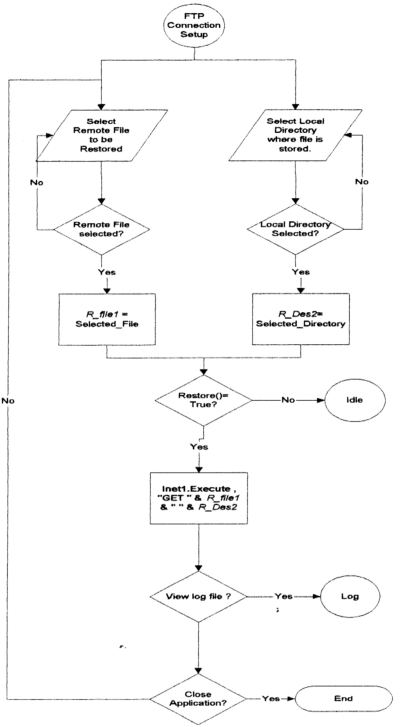


Figure 5.5 Remote Data Restore Sub-Module Flow Chart

### 5.3.3 SCHEDULED AND INCREMENTAL BACKUP MODULE

This module provides scheduled and incremental backup options to the local and remote backup system. Timer control and attributes property in visual basic can be utilized to perform scheduled and incremental backup respectively.

#### 5.3.3.1 SCHEDULED BACKUP SUB-MODULE

As a backup option that applies to both of the local and remote data backup module, this module will perform data backup in a scheduled and automated manner. A user is given the choice to select the appropriate days and time to do a backup or to perform a data backup daily at a specified time. Furthermore, the user can also determine the appropriate time interval for every backup in order to backup data frequently.

**Timer** event occurs when a preset interval for a **Timer** control has elapsed. The interval's frequency is stored in the control's **Interval** property, which specifies the length of time in milliseconds. Hence, local and remote backup modules can be invoked regularly under the timer control. Figure 5.6 illustrates the state diagram for scheduled backup.

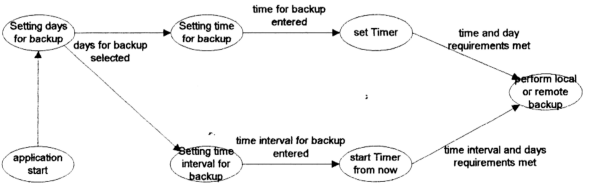


Figure 5.6 State Diagram for Scheduled Backup

### 5.3.3.2 INCREMENTAL BACKUP SUB-MODULE

Hence as illustrated in Figure 5.7 State diagram for incremental backup, this sub-module will perform data backup in an incremental manner. Therefore, with incremental backup option, backup module will only invoke when the file has changed since the last backup. User is given the choice to select whether performing backup incrementally or not.

As specified by Visual Basic for Application, each file has a specified attribute with a constant value. The two functions provided to implement file attributes are **GetAttr** and **SetAttr**. This two function may perform incremental backup by utilizing the **VbArchive** constant, to check if file has changed since last backup. Table 5.3 depicts the settings of file attributes argument.

Table 5.3 Settings of File Attributes Argument [49]

| Constant | Value | Description |
|---|---|---|
| **VbNormal** | 0 | Normal. |
| **VbReadOnly** | 1 | Read-only. |
| **VbHidden** | 2 | Hidden. |
| **VbSystem** | 4 | System file. |
| **VbDirectory** | 16 | Directory or folder. |
| **VbArchive** | 32 | File has changed since last backup. |

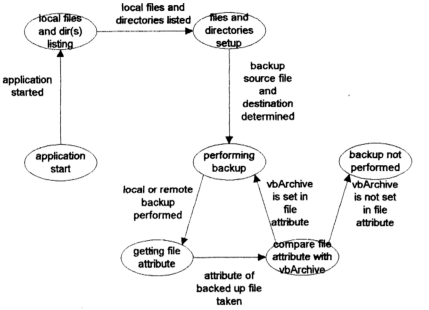


Figure 5.7 State Diagram for Incremental Backup

### 5.3.4 LOG FILES MODULE

This module provides documentation of the performed procedures in local and remote data backup system. Moreover, the progress of transferring source files to destination will be shown in the status window. The flow chart of this module is shown in Figure 5.8. Yet, it reports the file transfer progress for three modules, they are:

i) local data backup module
ii) remote data backup module
iii) remote data restore module

The log files in local and remote backup system are:

i) Log local_backup.txt
ii) Log remote_backup.txt
iii) Log remote_restore.txt

Accordingly, progress of data backup or data restore can be written to the specified files by using the **Open** statement. Then, **Print #** statement will write display-formatted data to the sequential file.

Statement: **Open** [48]

Syntax: **Open** *pathname* **For** *mode* [**Access** *access*] [*lock*] **As** [#]*filenumber* [**Len**=*reclength*]

| | |
|---|---|
| *pathname* | String expression that specifies a file name. |
| *mode* | Keyword specifying the file mode: **Append**, **Binary**, **Input**, **Output**, or **Random**. |
| *access* | Keyword specifying the operations permitted on the open file: **Read**, **Write**, or **Read Write**. |
| *lock* | Keyword specifying the operations restricted on the open file by other processes: **Shared**, **Lock Read**, **Lock Write**, and **Lock Read Write**. |
| *filenumber* | A valid file number in the range 1 to 511, inclusive. |
| *reclength* | Number less than or equal to 32,767 (bytes). For files opened for random access, this value is the record length. For sequential files, this value is the number of characters buffered. |

Statement: **Print #** [48]

Syntax: **Print #***filenumber*, [*outputlist*]

| | |
|---|---|
| *filenumber* | Any valid file number. |
| *outputlist* | Expression or list of expressions to print. |

After completing the file transfer, a report will be printed to a specified text file. User may view the log file in text format by executing the **ShellExecute** function. Initially, a declaration has to make in order to call this function from Visual Basic.

Syntax for **ShellExecute** function declaration [49]:

*Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" _*
*(ByVal hwnd As Long, ByVal lpszOp As String, ByVal lpszFile As String, _*
*ByVal lpszParams As String, ByVal LpszDir As String, ByVal FsShowCmd As Long) _*
*As Long*

Table 5.4 Descriptions for each Parameter in **ShellExecute** Function [49]

| **Parameter** | **Description** |
|---|---|
| Hwnd | Identifies the parent window. This window receives any message boxes an application produces (for example, for error reporting). |
| LpszOp | Points to a null-terminated string specifying the operation to perform. This string can be "open" or "print." If this parameter is NULL, "open" is the default value. |
| LpszFile | Points to a null-terminated string specifying the file to open. |
| LpszParams | Points to a null-terminated string specifying parameters passed to the application when the lpszFile parameter specifies an executable file. If lpszFile points to a string specifying a document file, this parameter is NULL. |
| LpszDir | Points to a null-terminated string specifying the default directory. |
| FsShowCmd | Specifies whether the application window is to be shown when the application is opened. |

In addition, the **ShellExecute** functions used to open these log files are:

i) **ShellExecute** hwnd, "open", WindowsDir & "Log local_backup.txt", vbNullString, vbNullString, SW_SHOW

ii) **ShellExecute** hwnd, "open", WindowsDir & "Log remote_backup.txt", vbNullString, vbNullString, SW_SHOW

iii) **ShellExecute** hwnd, "open", WindowsDir & "Log remote_restore.txt", vbNullString, vbNullString, SW_SHOW

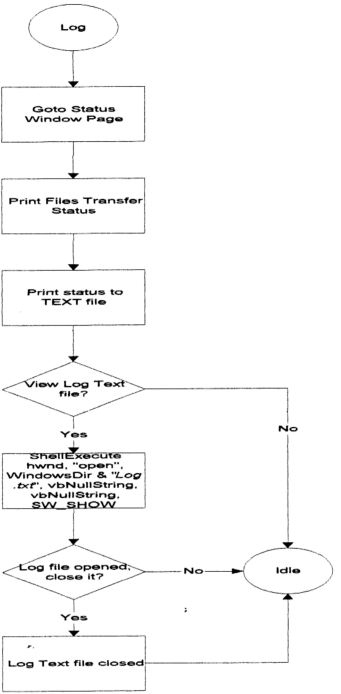


Figure 5.8 Log Files Module Flow Chart