Journal of Mobile Multimedia, Vol. 1, No.1 (2005) 033-046

AN IMPLEMENTATION OF LOCATION-AWARE MULTIMEDIA INFORMATION DOWNLOAD TO MOBILE SYSTEM

PHILIP K. C. TSE, W.K. LAM, K.W. NG, CHRISTOPHER CHAN

The University of Hong Kong, Hong Kong SAR, China

philiptse@ieee.org

Received March 18, 2005

Current mobile devices can capture and receive multimedia objects via the network operators. There are not many application systems that handle multimedia objects existing in both computers and mobile devices. We have successfully developed an application system to directly deliver multimedia objects from computers at local exhibition booths to the multimedia-enabled mobile phones. In this paper, we describe our target application as the location-aware multimedia information system in an exhibition centre. We have used different techniques including file stripping, proxy delivery, image preprocessing, transcoding, and Bluetooth protocol in this system.

Key words: Mobile Multimedia, Wireless computing, Bluetooth, Location-aware, file striping, proxy delivery, transcoding.

1 Introduction

Many mobile network operators provide the downloading service to their subscribers. The mobile network operators store some applications, games, and ring tones on the company web sites. Users may then search the files that they wish and download from the site to the mobile phones. Apart from these files, mobile phones can also connect to computers in the vicinity using wireless communications. The mobile phones may then access files on the local computer.

Mobile phones can now take digital pictures and short videos. They can display images and videos with many colours onto the small screen. They can record audible sound from the microphone and output audio sound to a small speaker. They are multimedia enabled computing devices. New application systems should be able to handle multimedia information in the mobile phones.

In the existing downloading services, users need to manually perform file searching by frequently operating the small keys in the keypad. An automatic downloading system would be able to provide more convenient interface to users in accessing information.

Automatic downloading system can provide more appropriate information to the users if it considers the user location information. For example, user in a local exhibition booth may collect exhibitor ŝ information while visiting exhibition booths or user may collect local information while driving near a local visitor information center. Thus, automatic downloading system shall make use of the user location to select the suitable files to download.

We first describe the project objective and the development tools in the next Section. Then we describe our system implementation in the Section 3. After that, we discuss limitations and possible extensions to our system in Section 4. Lastly, this paper is concluded in Section 5.

2. Project Objective and Development Tools

In a simple file download system, a server only provides direct download service to the users. In the mobile environment, the user is likely to move around. The mobile device can easily move out of the physical range of one server and into the physical range of another server. This switching of mobile device connection from one server to another interrupts the downloading process. If the entire downloading service needs to restart from the beginning, the downloading process may never finish. It would be nice if the newly connected server can resume the downloading process by proxy delivery of the partially downloaded file.

In an exhibition centre, companies showcase their products at the exhibition booths. Direct communications between visitors and salespersons facilitates future purchases. Visitors move from booth-to-booth watching demonstrations and listening to explanations from salespersons. Most visitors have limited time to visit a booth, so the salesperson can only briefly demonstrate and describe their products. Salesperson may provide details information to the visitors who would like to take them home for future reference. The collection of leaflet that is received by a visitor is often unstructured and difficult to manage. In order to create a database of the information of the visited booths, the visitor should collect such information during his visit using a mobile device. Therefore, our objective is to develop a location-aware information system that can automatically download multimedia files to a J2ME-enabled mobile phone.

We have examined and tested a wide range of development tools for mobile phones, such as BlueZ, Nokia Developer ŝ Suite for the Java™2 Platform, Series 60 SDKs for Symbian OS, Borland JBuilder Personal Edition, and the Impronto Developer Kit.

Nokia Developer ŝ Suite for J2ME™ allows content creation for Mobile Information Device Profiles (MIDP) versions 1.0 and 2.0. It provides tools to create MIDlet classes, create application packages as MIDlet Suites, emulate and deploy MIDlets. The Borland JBuilder is a software for writing JAVA applications. It allows user to write JAVA codes, compile the program, and package the application in jar files. It integrates well with Nokia Developer ŝ Suite for J2ME so that the emulator can be invoked to run the program.

The Impronto Developer Kit that is developed by Rococo Software Limited is available free to university and hobbyist programmers for non-commercial use. It hides the complex Bluetooth protocols from the users who invokes the Bluetooth API (JSR82) in Linux. Thus, it allows us to create Bluetooth connections at a low development cost.

3. System Implementation

Our application, the Location-Aware Multimedia Information Download System (LAMIDS), consists of three major parts: the mobile phones, the servers and the File Location Register. The designated system architecture is shown in Figure 1. Our application may also be implemented on other system architectures in which computers are installed with Bluetooth adaptors. We have adopted the client-server model in this project.

At the server side, our application is a JAVA program that runs on the Linux platform. The implementation of Bluetooth communication is achieved by using the JAVA Bluetooth API in the Impronto Developer Kit for Linux. The servers, which are installed in different areas of the exhibition centre, can deliver the files to mobile phone. We use Pentium PCs at 700MHz with 256MB RAM and 2GB hard disk as the servers in our prototype implementation. A Bluetooth adaptor is connected to the

USB port. Each server stores different media files such as text files, images, music clips and short movie clips. All these servers are connected on a LAN (Local Area Network). While the phone user is staying close to different servers in the exhibition centre, different files are delivered to his mobile phone in an ambient manner.

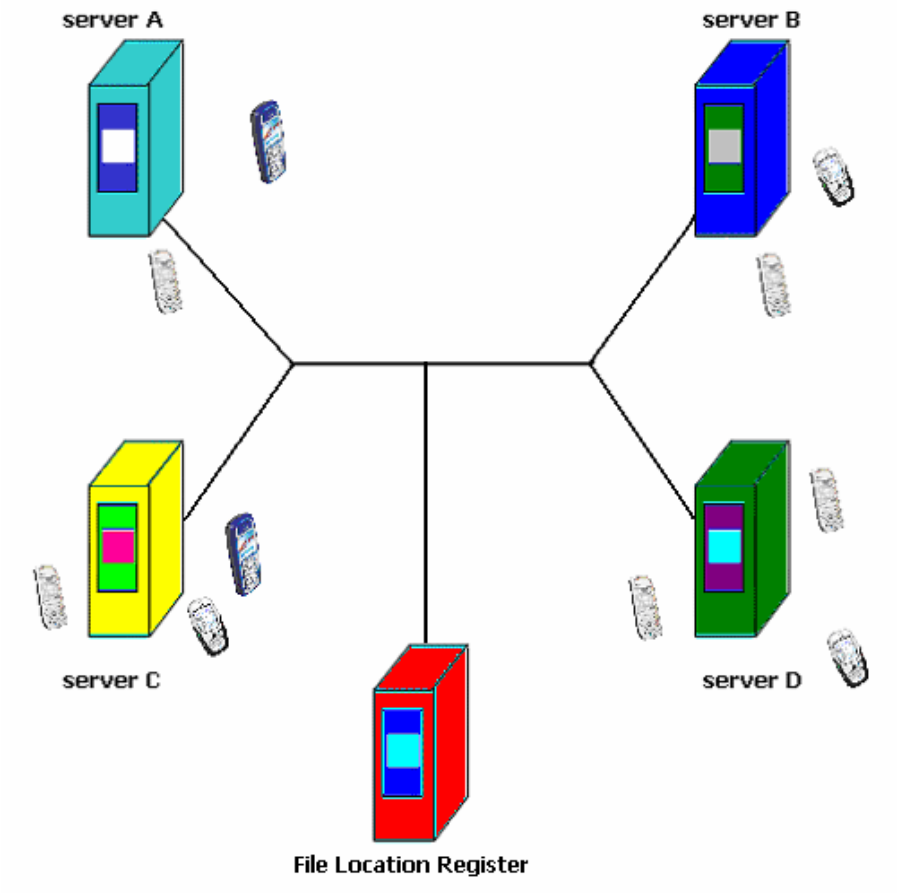


Figure 1 The system architecture

Besides, the servers preprocess the image files in advance. This feature does not only help to save the amount of storage in the phone, it also reduce the amount of time to download the image files under the limited bandwidth of Bluetooth connection [8]. Also, the server can deal with the proxy caching function from other remote servers. Therefore, the file transmission can continue as long as the phone connects to any servers in the exhibition centre.

At the client side, the application runs on a Nokia 6600 mobile phone. This mobile phone has the Bluetooth API (JSR-82) to establish communication with the computers. It supports the J2ME™ platform for application running. It has an internal memory of 6MB and a memory card of 32MB that are large enough for temporary multimedia object storage. It supports resolution of 176x208 pixels at 16 colours to display videos and images. A user-friendly graphical user interface is created to facilitate the operations of the application.

Furthermore, our application is cost-effectiveness to develop because our system is based on JAVA and Linux, which are both free for developers. Our information system can also be deployed in many public indoor areas, for example, an information enquiry system in a shopping mall. With this system, the information, such as the locations of different facilities in certain floor of shopping mall, can be used to guide the visitors.

We present the application and the data structure of media files in the mobile phone in Sections 3.1 and 3.2. Then in Sections 3.3 and 3.4, the delivery of media files to mobile phones and the proxy delivery function are described. Afterwards, we present the Administration Tool and the file preprocessing in Sections 3.5 and 3.6.

3.1 Application in the mobile phone

Application in the mobile phone is implemented by using J2ME MIDlet programming techniques and a few APIs. The J2ME, MIDP 2.0 API focuses on smaller subgroup of device families having the same characteristics and similar target application [4]. The Bluetooth JAVA API (JSR-82) is used to connect the phone with the computers. The Mobile Media API (JSR-135) is used to play the multimedia files. The program structure of the application looks like a tree in Figure 2.

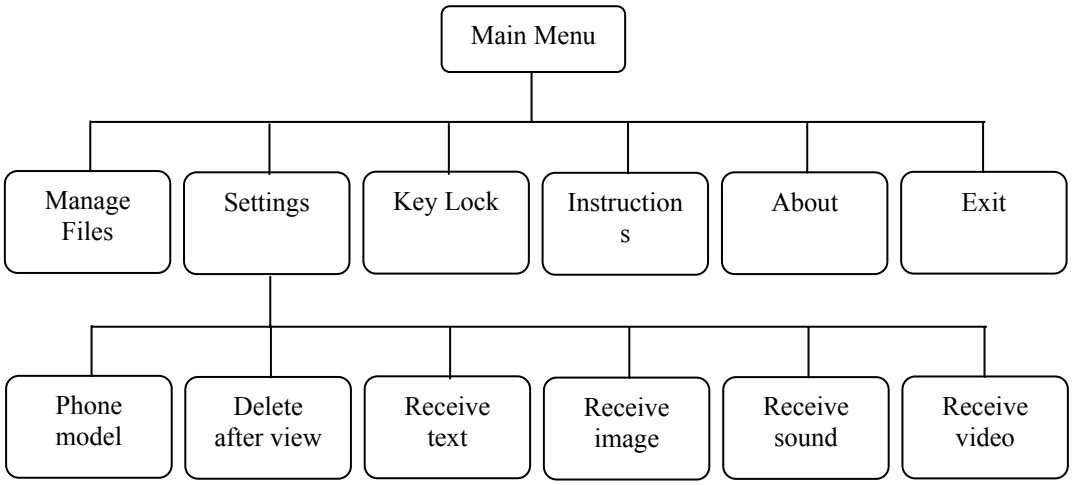


Figure 2 Tree structure of application in the mobile phone.

3.1.1 Setup Connections

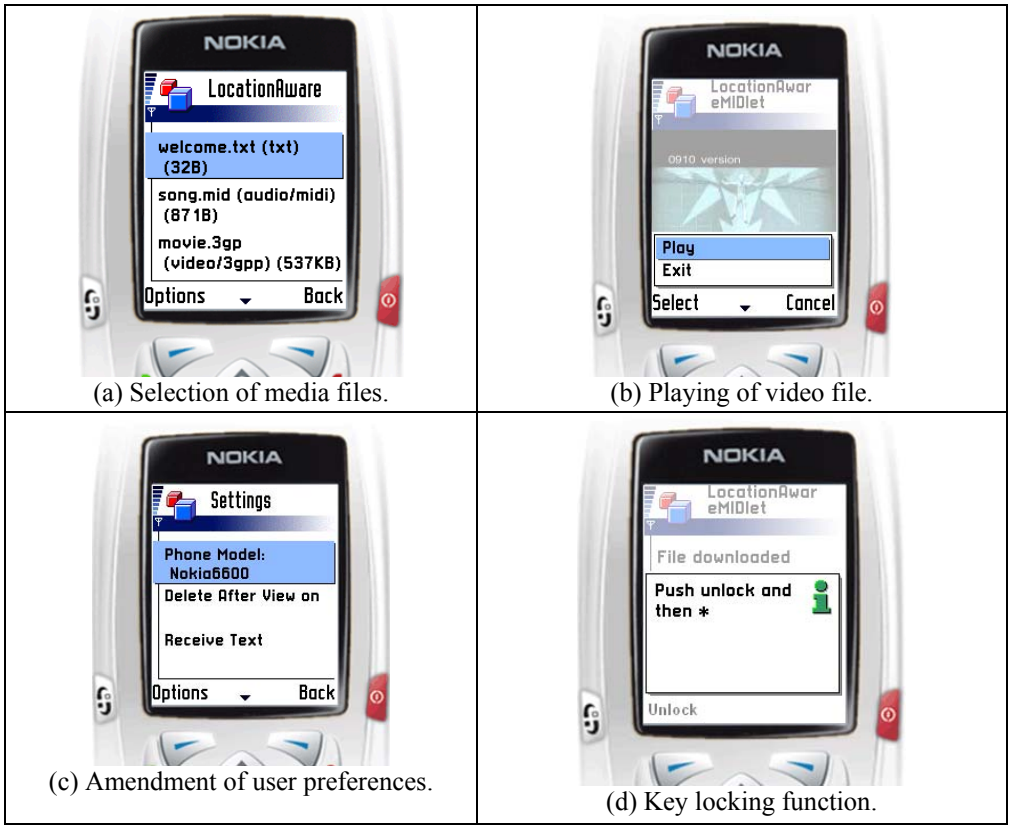
After the user invokes the application in the phone, the application takes the initiative to establish the communication link with a nearby server. One of the connected servers will become the “ local ” server and all other servers will become “ remote ”servers.

When the application is launched in the phone the first time, the user will receive only the default files delivered by the server automatically. The user will also not be interrupted by the downloading service, as a complete download will not alert the user. During downloading of file(s), the user can cancel the file transmission at any time whenever the user wants to.

When there is incoming phone call, the phone notifies the server to pause the data transmission. After the incoming call is finished, resumption of transmission will be initiated automatically. This ensures that data files in the phone are not corrupted.

3.1.2 File Management

This application is built to perform automatic downloading. A simple graphical user interfaces is created for users to easily access to any functions in the mobile phone as shown in Figure 3.



(a) Selection of media files.

(b) Playing of video file.

(c) Amendment of user preferences.

(d) Key locking function.

Figure 3 Functions available in the mobile phone. (All figures are captured from the emulator.)

After the multimedia files have been downloaded, the application will store the files in the phone memory so that the user can access the files. After the files are received completely, they can be viewed or played. The user can have a look at the list of the files downloaded (Figure 3a) and choose whether to play it or not (Figure 3b).

3.1.3 Setting User Preferences

The easy-to-use graphical interface will show the status of file delivery and let the user modify six preference settings in the phone. The user should specify the model of the phone (Figure 3c). The servers will only transfer the files that are supported by the phone model. For example, the server will not transfer a large image file to a phone that cannot display it on its small screen. The administrator of the system will specify the screen size of each phone model for comparison.

The user can choose whether the viewed files should be deleted or not. If this is set to yes, the files will be immediately deleted after viewing. The playing of a multimedia file is allowed after a sufficient large proportion of the file has been downloaded. If the capacity drops below a threshold value, the application will automatically erase the files with the lowest priorities to release space. Alternatively, the user may manually erase the downloaded files.

He may specify whether to receive the text files, the image files, the sound files, and the video files or not. The files to transfer will then be filtered by the specified receive file types. For example, a user may change the receive sound setting and the receive video setting to no. Then the application will transfer text files and images files only.

3.1.4 Key Lock

It is assumed that the user will turn on the application and leave it alone to download multimedia files itself. So, we have implemented a key lock function within our client application so that pressing buttons mistakenly would not hinder the delivery of files (Figure 3d).

3.2 Data structure of the media files in the mobile phone

Because of the security feature of JAVA, so called "Sand Box model" [9], the mobile phones currently prohibit access to their file systems. So, RecordStore is used to store files inside the phone. RecordStore, which is just like a database, is being used in mobile phone. We have used RecordStore to create a file structure for the permanent data storage in the phone (Figure 4). A class is created to operate the files, such as the storage of file content, the retrieval of file content, the deletion of file content, and the appending of file content.

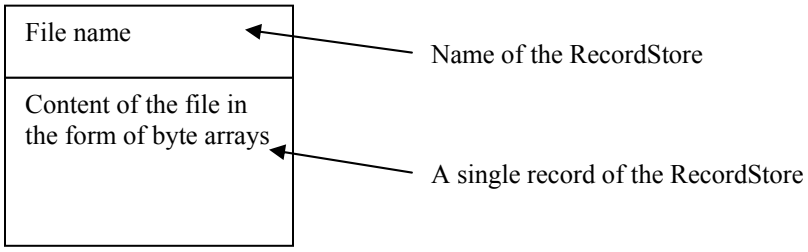


Figure 4 The architecture of File by using RecordStore

The creation of file using RecordStore is quite simple. Each file is stored in a corresponding RecordStore using the name of the file as the name of the RecordStore. Then the record in that RecordStore will store the byte arrays of the file content. This approach is straightforward to permanently store data in the mobile phone. As a result, the files can only be accessed within our application and we have to develop a user interface to view and play the stored files.

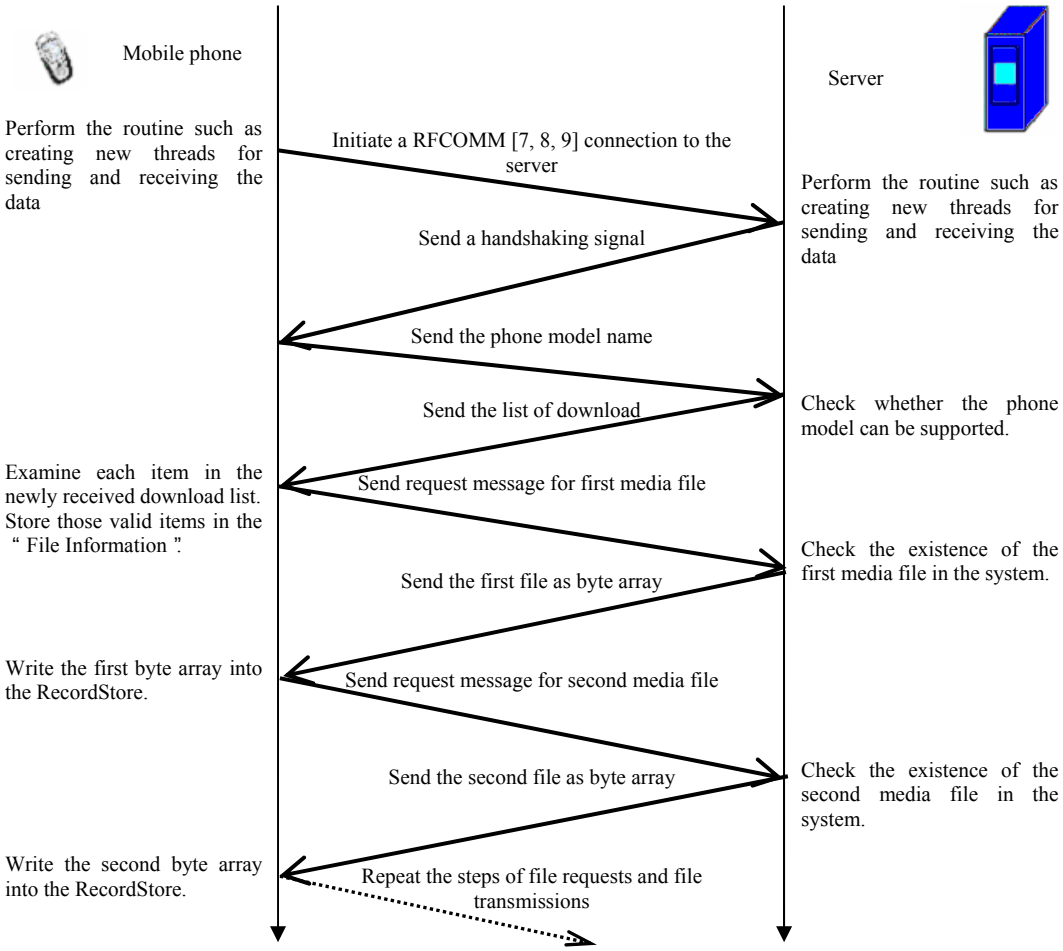


Figure 5 Information Exchange during automatic downloading [2]

3.3 Delivery of media files to mobile phones

The server should wait for any incoming Bluetooth connections from mobile phone. After the establishment of connection, the server should automatically:

1. Deliver the list of download which matches the phone model;
2. Transmit the media files from the server to the phone;
3. Cache any media files from remote servers, if necessary.

The detailed procedures of automatic downloading are shown in Figure 5.

Besides, if the requested media file does not exist in the system, then the server will send a signal to force the mobile phone to delete the concerned record of file information inside the mobile phone.

Finally, in order to enable the mobile phone to automatically search for the nearest server, the user can change the security settings to permit the JAVA application to automatically access to the Bluetooth function.

3.4 Proxy delivery function with file stripping

From the view of phone user, if his / her mobile phone has direct connection with the server, this phone user will regard that server as " local server "and all other servers as " remote servers. "Proxy delivery function is invoked when the record of the target media file cannot be retrieved from the local server. In this case, the local server is required to send a query to the File Location Register. A File Location Register is a server storing the locations of all media files existing in any servers belonging to the system.

After retrieving the query result from the File Location Register, the local server will know where the target media file is stored. It will simultaneously send proxy caching requests to all " Online " remote servers having the copies of the target media file. If there is more than one remote server having the same copy of the target file, the caching with " file stripping "can be achieved by receiving the file strips of the target file from the corresponding remote servers. After completely receiving all file strips from the remote servers, the local server combines all these file strips with the desired sequence. Then the local server will deliver the resultant media file to the mobile phone.

Besides, another condition of caching of the file is that the locally cached media file is not the most updated one, comparing with the original copy in the remote server. For the old version of the file, no related records can be found in the File Location Register. Therefore, the local server can examine the version of the media file through the File Location Register. If the local server finds that the cached object is not the most updated version, it will then cache the most updated version of media file.

3.5 Administration Tool for content administrators

We have built an Administration Tool to manage the media files in the server. With easy-to-use GUIs in the Administration Tool, the content administrator can effectively manage the media files in the server. Besides, the content administrator is recommended to add, update, or remove the media files through Administration Tool only. The administrator should not use other application programs to manually change any files inside the default directories storing these multimedia files.

Apart from the management of media files, the status of the server can be directly examined by looking at the title bar of the Administration Tool window. Also, the Administration Tool allows the content administrator to perform other administration functions, such as change passwords, manage phone models, manage download list for different phone models, configure server settings, and view activity logs. We shall describe these functions below.

3.5.1 GUI of Administration Tools

We have used JAVA Abstract Window Toolkit (AWT) and the JAVA Swing to construct the GUIs of Administration Tools. Most components, such as buttons, textboxes, and menu bars, are constructed using JAVA Swing. In addition, JAVA AWT is used to construct the layout and define actions for the event handling. The components and the related actions of those components are implemented inside the files " UIMain.java ", " DatabaseManipulations.java " and " FileManipulations.java ". The " DatabaseManipulations.java "file is used to maintain the records in the local server and the records in the File Location Register, whereas the " FileManipulations.java " file is used to add and delete media files.

The server name and the status of the Bluetooth adapter in the server side are shown on the title bar of the Administration Tool window (Figure 6). Before the system detects any Bluetooth adapters on the server side successfully, it displays the keyword "Offline" in the title to indicate that the server cannot accept any incoming requests from any phone users. After the Bluetooth adapter is ready, it shows "Online" in the title bar.

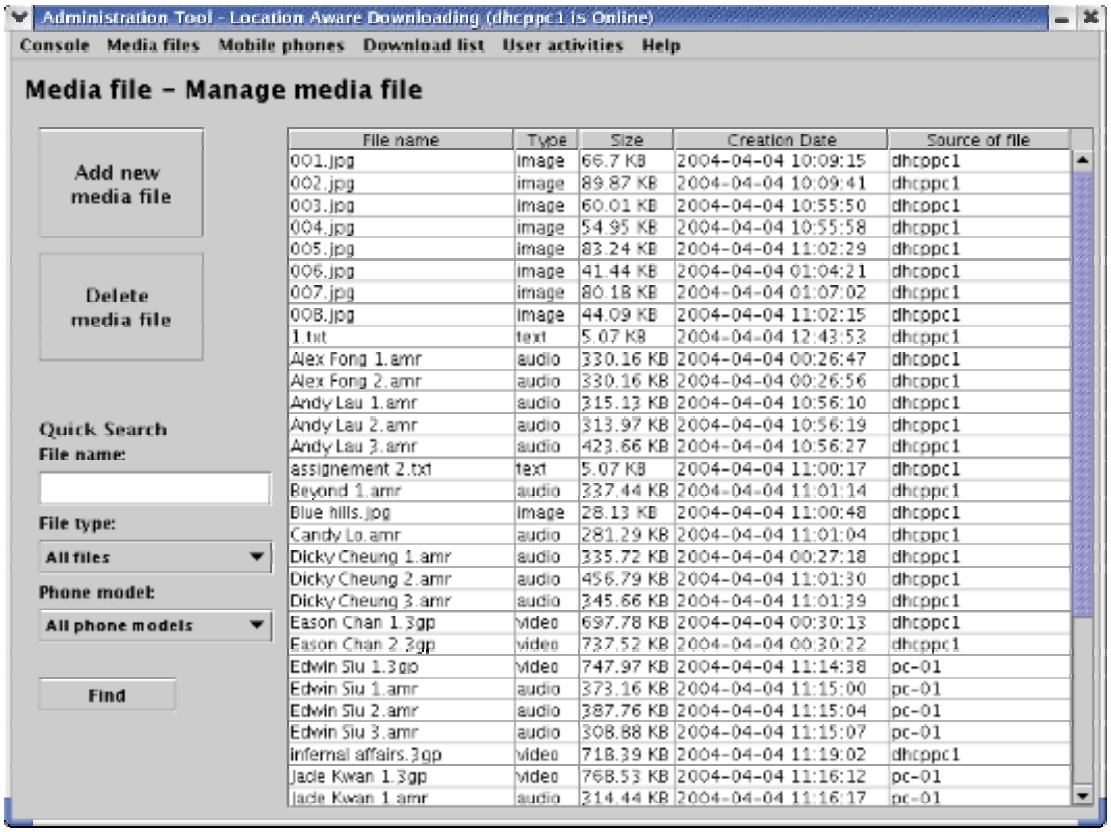


Figure 6 The user interface for the management of media files stored in the server

3.5.2 Authentication

To authenticate the content administrator, a standard login interface has been implemented. Initially, there is an interface requesting the content administrator to input the user name and the password. If the local database verifies that the user name matches with the input password, a main menu is displayed. Otherwise, a warning message is shown.

3.5.3 Main menu

Under the main menu, the content administrator can select the desired functions from menu bar directly, for example, changing the password or managing the media files.

3.5.4 Change password

To change passwords, one extra textbox is created for content administrator to retype the new password and the system will double check the correctness of the new passwords.

3.5.5 Manage media files

We have created one GUI to manage the media files, including adding files, deleting files and searching a specific file (Figure 6). The content administrator only applies a few mouse clicks to certain buttons. If there is no media files stored under the directory "media", then a message will be generated to remind the content administrator on the way of adding media files.

The system will show the detailed file attributes, including file name, file category, size, creation date, and the source of file. The player in the phone supports four media file types as shown in Table 1. The "Source of file" shows the server holding the file. If this field is not same as the hostname of the "Local Server" displayed at the title of the window, the media file is cached automatically by proxy delivery instead of added manually by the content administrator.

Table 1 List of file types supported by the player in the phone

File Category	**Supported file types**
Text	Txt
Image	bmp, png, jpg, jpeg, gif, tif, tiff
Audio	audio/midi, audio/x-tone-seq, audio/x-wav, audio/au, audio/basic, audio/amr, audio/amr-wb, audio/sp-midi
Video	video/3gpp, video/mp4, application/vnd.rn-realmedia

For the addition of file, a JAVA swing component called File Chooser is used to implement the selection of file. This File Chooser can filter away any media files types that are incompatible with the media player and the viewer in the mobile phone.

When any media file is added, the Administration Tool checks for the validity of:

1. Duplication of file name. If the file name of that media file already exists in the "Local Server", the Administration Tool will consider this case as the update of media file. A message is generated to ask the content administrator whether the concerned media file should be updated.
2. Incompatibility of file type. If a media file is incompatible with the viewer / player in the phone, a warning message is displayed to notify the content administrator.

After an image file is added to the server, preprocessing is carried out on the new image file in order to reduce the transmission time and storage space in the mobile phone.

The administrator may delete media files by selecting the files from the file list and clicking the "Delete" button. Before deleting any media files from the local drive, a warning message is required to ask for confirmation. This can avoid the accidental deletion of media files.

3.5.6 Quick Search

The “Quick Search”function is also provided to search any local media files. The result of file list will be generated according to the keywords of file name, the selected file type and the selected phone model. If the administrator selects a particular phone model in the “phone model”field, the image files that are incompatible with the phone model will be filtered away from the file list. The selection of phone model in “Quick Search”is very convenient in finding specific files.

3.5.7 Management of phone models

Apart from the management of media files, a GUI to manage the list of registered phone models is also implemented in the Administration Tool. In this GUI, all supported phone models are shown in the phone model list. When any phone models are added or updated, warning messages will be generated for invalid data type and the phone model name duplication. When any phone model name is selected by the mouse click, the corresponding detailed phone attributes is displayed. If the administrator wants to remove any phone models, he / she can directly clicks the button “Unregister phone”to delete the phone model. After the phone models and media files are added into the server, the content administrator can create the download file list.

3.5.8 Management of download list

The Administration Tool also provides a GUI to manage the list of files to download for a phone model. A media file list and a download file list are displayed to the administrator. The content administrator may select a phone model and the files that are incompatible with the selected phone model will be filtered away from the media file list. The administrator may select the media files and move them to the download file list. The download file list is ordered according to the download priority. The administrator may also change the priorities of the download files.

3.5.9 Server settings

A GUI to save the server settings, such as the IP address of the File Location Register and the path of storing the directory “media”, is implemented in the Administration Tool. This GUI will be invoked when the server cannot connect with the File Location Register or the path of storing the media file is not defined. After the content administrator logins to the Administration Tool successfully, he / she needs to input the valid information in order to activate the server. If the IP address of the File Location Register is invalid, an error message will be generated.

3.5.10 Statistics and user activities

A GUI is implemented for the content administrator to view the statistics, for example, the total number of visits. Another GUI is implemented for the content administrator to view the activities performed by certain phone user.

3.6 Preprocessing of image files

The image files used in this project need to be preprocessed by the servers. Files with too many pixels or too many colours are undesirable to transmit because the mobile phones cannot display with such details. With image preprocessing, unnecessary data transmissions can be avoided and the transmission bandwidth can be better utilized.

As different mobile phones may provide different display characteristics, the image files are converted into several different versions according to the phone models. Each mobile phone can then choose the version that best suits its functionalities. In this project, preprocessing is performed automatically in the management of media file in the Administration Tool. Therefore, when the administrator adds any image files to the server, the resolutions of the image files will be decreased and the resultant files will be created and stored accordingly.

In this project, transcoding is done mainly on reducing the file size; there is no doubt that the algorithm used must be a lossy compression algorithm. That means, once an image file is transcoded, the original source image cannot be fully recovered. Therefore, the original file is also added to the database to store the image source.

One generic algorithm to implement this feature is to perform a direct downsampling. This algorithm is simple to implement and its computation time complexity is linear to the size of the pixel matrix. Unfortunately, the loss of information is greater than some other implementations.

An alternative algorithm is to filter the pixel matrix. Filter operation is good in the sense that although it is also a lossy implementation, it will consider neighboring correlated pixel values during manipulations. This operation is image-independent and depends on the implementation to make it more flexible. Also, as correlated pixels are concerned, the JPEG compression algorithm used afterwards will yield a larger compression ratio, i.e. reducing the file size further. However, the computation time is high. The overall time complexity is found in [10] as

$$O(n^2) = (\text{size of pixel matrix}) \times (\text{size of filter}) \times (\text{no. of channels of colour model})$$

The two different algorithms are attempted on some image files and their corresponding performances have been compared. It is found that the simple downsampling algorithm can already produce acceptable visual effects, comparing to the mean-filter algorithm. Although the latter algorithm produces a higher compression ratio, its operations are much more complicated, both in implementation and computation time. After considering the computation time at the server, we thus use only the simple downsampling algorithm in this application.

4. Discussions

We have successfully developed a location-aware downloading system that can automatically download multimedia files to a J2ME-enabled mobile phone via Bluetooth connection. This system provides convenient interface to users in accessing information and it downloads files automatically. It uses the user location information to select the suitable files to download. After the interruption to a downloading process is recovered, the downloading process will resume using proxy delivery of the partially downloaded file. The mobile phone can play multimedia files after downloading. We shall describe the limitations that we have encountered below.

Up to the current stage, this location-aware information system only supports J2ME-enabled mobile phones with Bluetooth connectivity. As only eight devices can be supported in the same Bluetooth network, the efficiency of the server loading in our system is rather limited. Besides, the bandwidth of the Bluetooth communication is only limited to 128 kbps when the serial port profile is adopted [7]. Since only serial port profile is available in JAVA Bluetooth library in the application level, it is found that the speed of file transfer is limited by this serial port profile. Also, the

computation speed of the mobile phone will affect the downloading speed of video file. From our experiments, it usually takes around 5 minutes to download a 700 KB video file to the mobile phone.

Apart from the limitation of Bluetooth connectivity, there is also limitation on the implementation of video streaming over Bluetooth. The Player Class in Mobile Media API (JSR-135) can only perform streaming through some common protocols, e.g. RTP streaming, which is based on TCP connection. Unless we establish a TCP connection, the video streaming cannot be performed directly through Bluetooth using the Mobile Media API. It is interesting to implement TCP connection over Bluetooth connections in future enhancements when real-time streaming is needed.

For the image preprocessing [3], movie clips may also be compressed to smaller file size. Recently, the 3gpp file format is one popular movie file type being used in mobile devices. This file type provides a much better compression performance and consumes less storage space.

In order to provide the functionalities of this multimedia information system to a wide area, the mobile phone may connect via the 3G system, which is recently available in Hong Kong. With a much higher transmission rate in the 3G system [11], real-time streaming of multimedia content becomes possible and storage of audio and video files could then be avoided. In the 3G system, each base station has its unique identification, the mobile phone can then distinguish its current location with such identification. With the ability of distinguishing the location, the location-based application can be implemented onto a wider area of coverage.

Although this system may be enhanced in download speed, real-time streaming, wide area location identification, and higher compression, our current system is already a feasible system to provide location-aware multimedia download services within an exhibition centre environment.

Our system downloads information from servers to mobile phones. The system can also be applied to other mobile environments. It is directly applicable to download information to any mobile devices that supports the J2ME platform. It is also applicable and easier to upload information from mobile phones to servers. The local server may also deliver media files from other remote servers on the Internet to the mobile phones. If servers on moving vehicles are connected using wireless communication, travelers on these vehicles may also exchange their files via the servers.

5. Conclusion

We have built a location-aware multimedia information download system for mobile phones. In this system, the content administrator can effectively manage the media files, control the list of downloadable files, and observe user activities. The system downloads the most appropriate set of files according to the user location information, file types, and the phone models.

Once the application is started in the mobile phone, multimedia files will be transferred in an ambient manner to the mobile phones. The user can pay attention to other matters. Such feature can also be applied when the user is driving a car and the mobile phone may collect information from local visiting points without disturbing the user. The application is also user-friendly to use that it locks keys during download, resumes download after interruptions, and automatically manages storage space by removing low priority files.

We have used different techniques including file stripping, proxy delivery, image preprocessing, transcoding, and Bluetooth protocol in building this system. It is a cost-effective and flexible solution to download multimedia information to mobile devices in a mobile environment.

Acknowledgements

We would like to thank Dr. Vincent Tam (Department of EEE, The University of Hong Kong) and Dr. K.W. Chong (Department of CSIS, The University of Hong Kong) for their valuable advices and comments throughout this project. Philip Tse's research is supported by the Areas of Excellence Scheme established under the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E/01/99).

References

1. Chan, C.L.S., Entertainment Download Exchange for Mobile Devices □ Project Report, FYP Technical Report, Department of EEE, HKU, 2004.
2. Lam, W. K., Entertainment Download Exchange for Mobile Devices □ Project Report, FYP Technical Report, Department of EEE, HKU, 2004.
3. Ng, K.W., Entertainment Download Exchange for Mobile Devices □ Project Report, FYP Technical Report, Department of EEE, HKU, 2004.
4. Haque, I., O'Connor, B., J2ME Enterprise Development, M & T Books, New York, 1st edition, 2002.
5. Hopkins, B., Antony, R., Bluetooth for Java, Apress, New York, 1st edition, 2003.
6. Kumar, C. B., Kline, P. J., Thompson, T. J., Bluetooth Application Programming with the Java APIs, Morgan Kaufmann, San Francisco, 1st edition, 2004.
7. Mullar, N. J., Bluetooth Demystified, McGraw-Hill, New York, 1st edition, 2001.
8. Forum Nokia □ Developer Resources: http://www.forum.nokia.com/
9. Forum Nokia □ Developer Discussion Board: http://discussion.forum.nokia.com/
10. Sharma, R., Sharma, V., Java Programming by Example, Cambridge University Press, 1st edition, 1998.
11. Muratore, F., UMTS Mobile Communications for the Future, John Wiley, New York, 2001.