# Personal information search on mobile devices

Akbas, Mehmet.

Monterey, California. Naval Postgraduate School


# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**PERSONAL INFORMATION SEARCH ON MOBILE DEVICES**

by

Mehmet Akbas

September 2007

| | |
|---|---|
| Thesis Advisor: | Gurminder Singh |
| Co-Advisor: | Thomas Otani |

**Approved for public release; distribution is unlimited.**

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | | Form Approved OMB No. 0704-0188 |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503. | | | |
| **1. AGENCY USE ONLY** ***(Leave blank)*** | **2. REPORT DATE** September 2007 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis | |
| **4. TITLE AND SUBTITLE** Personal Information Search on Mobile Devices | | **5. FUNDING NUMBERS** | |
| **6. AUTHOR(S)** Mehmet AKBAS | | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | **8. PERFORMING ORGANIZATION REPORT NUMBER** | |
| **9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** | |
| **11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. | | | |
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited. | | **12b. DISTRIBUTION CODE** | |


**13. ABSTRACT (maximum 200 words)**

Today's mobile devices, especially mobile phones, are comparable in computing capability and storage to the desktop computers of a few years ago. The volume and diversity of the information kept on mobile devices has continually increased and users have taken advantage of this. Since information is being stored on multiple devices, searching for and retrieving the desired information has become an important function.

This thesis focuses on search with regard to Personal Information Management (PIM) on mobile devices. A search system which involves different types of mobile devices is also introduced.


| **14. SUBJECT TERMS** Search, PIM, mobile device, information management | | | **15. NUMBER OF PAGES** 103 |
|---|---|---|---|
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UU |

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited.**

**PERSONAL INFORMATION SEARCH ON MOBILE DEVICES**

Mehmet Akbas
Major, Turkish Army
B.S., Turkish Military Academy, 1992

Submitted in partial fulfillment of the requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL**
**September 2007**

Author: Mehmet Akbas

Approved by: Gurminder Singh
Thesis Advisor

Thomas Otani
Co-Advisor

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

## ABSTRACT

Today’s mobile devices, especially mobile phones, are comparable in computing capability and storage to the desktop computers of a few years ago. The volume and diversity of the information kept on mobile devices has continually increased and users have taken advantage of this. Since information is being stored on multiple devices, searching for and retrieving the desired information has become an important function.

This thesis focuses on search with regard to Personal Information Management (PIM) on mobile devices. A search system which involves different types of mobile devices is also introduced.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

# LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

## ACKNOWLEDGMENTS

I would like to dedicate this thesis to my two sons, Kagan and Cagan. They have always been ready to play when it was time to have a break.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. PROBLEM SPACE

A large number of people today use multiple devices to accomplish their jobs. It is quite common for an individual to be using a smart phone, a laptop computer, a USB storage device and a desktop computer, and in many instances more than one of each type of device is used. In recent years, the power and capacity of these devices has become quite significant and hence people are beginning to store and use their information across all these devices. For example, mobile phones which previously only stored contact lists are now used for storing pictures, music, video clips, calendars, task lists and notes.

With their personal information scattered among so many different devices, it has become difficult for mobile users to manage, find and use this information in an efficient way. Some contacts are saved on the PC in an address book, some email addresses are only in the online email account's contact list and some phone numbers are stored only in the mobile phone's memory. Personal Information Management (PIM) addresses these problems. Teevan et al. (2006) defines PIM as [1] *"Personal information management (PIM) is intended to support the activities we, as individuals, perform to order our daily lives through the acquisition, organization, maintenance, retrieval, and sharing of information."*

Today's mobile devices, especially mobile phones, are comparable in computing capability and storage to the desktop computers of a few years ago. The volume and

diversity of the information kept on mobile devices has continually increased and users have taken advantage of this. Consequently, searching for and retrieving the desired information on mobile devices has been become an important function. Increasing number of users and businesses find themselves more heavily dependent on mobile devices than ever before. As a result, managing information which is scattered across multiple devices is becoming a big problem. In the absence of proper tools, searching for and retrieving personal information can end up being a digging-everything-inside-out expedition.

## B. OBJECTIVE

The objective of this thesis is to provide an understanding of search issues with regard to PIM on mobile devices and develop tools for efficiently and conveniently searching through personal information stored on mobile devices.

## C. RESEARCH METHODOLOGY

The research is comprised of three parts. First, PIM and its subcomponents will be evaluated. Second, mobile devices and their capabilities along with the connection techniques with each other will be discussed. Finally, a demonstration application will be presented.

Any useful and convenient search system for personal information must cover mobile phones, PDAs, personal computers and the personal data residing on the Internet. The demonstration application included in this thesis does exactly this: it is a search system that facilitates finding

personal information dispersed over multiple devices including mobile phones, personal computers and finally, the internet.

## D. THESIS ORGANIZATION

The remainder of the thesis is organized as follows. Chapter II provides an overview of PIM for mobility. Search issues in PIM and sub components of PIM will also be evaluated in this chapter.

Chapter III describes mobile devices and their capabilities. It introduces and provides an overview of abilities that current mobile devices offer with comparison to computers and previous mobile devices. Also in this chapter, the connection techniques and schemes including Bluetooth, GPRS, Wi-Fi will be discussed.

Chapter IV is devoted to a demonstration application which implements search function on multiple mobile devices. The design and implementation of the application will be discussed in detail. The observations and application usage scenarios will also be explained in this chapter.

Finally, Chapter V summarizes the entire study and presents conclusions reached concerning search issues and feasibility of applications demonstrated. Additionally, Chapter V discusses recommendations for future research in this area.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. OVERVIEW

Personal Information Management (PIM) describes the acquisition, organization, and retrieval of information by an individual user [2]. The phrase “personal information management” was first used in the 1980s when almost everyone expected too many things from the newly popular device of that time -the personal computer. Among the expectations was the huge improvement of the human ability to process and manage information [3].

In this chapter, PIM and its subcomponents will be discussed, and search, which is an important element of PIM, will be elaborated.

## A. OVERVIEW

PIM helps users access more information while eliminating the risk of losing what is important at the same time. As the cost of digital storage becomes lower and the availability higher, more and more information is being stored in mobile devices. But keeping everything and storing every bit of information have obviously negative side effects such as information overload and distracted attention [4]. In many cases, a user looks for specific information in this enormous pile of information. Without proper tools and techniques it would be a time-consuming, error-prone activity which can lead to completely unsatisfactory result.

Better search support in PIM enables users to pinpoint the desired information even when it is hidden in immense accumulation of irrelevant information in different device

databases. Better PIM and better tools and techniques let users make better use of invaluable time.

## B. PIM FOR MOBILITY

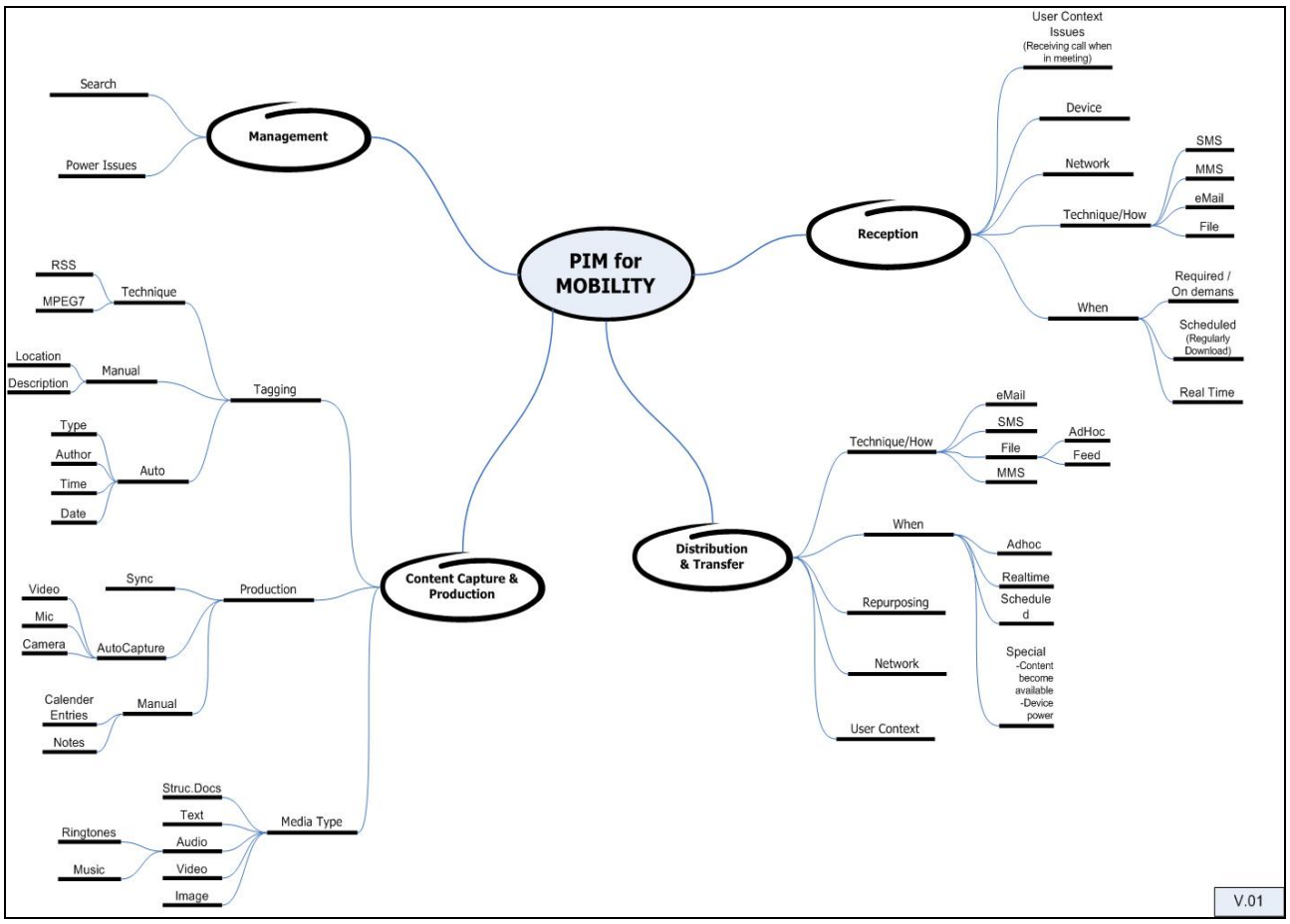


Figure 1. Taxonomy of PIM for Mobility.

### 1. Introduction to Mobile Devices

Traditionally, a mobile device is defined as a pocket-sized computer and a communication tool which has a tiny display screen and a miniaturized keyboard or touch-screen interface for user input. Nowadays, mobility and wireless communication are supported by many other types of devices such as digital cameras, video recorders, watches even comes

with the form of clothing. With a broader view, mobile devices include descendents of PCs, PDAs, mobile phones, PDA/mobile phone combinations, beepers and pagers, wearable computers, watches/jewelry and cameras [6].

Portability, light weight, integrity and most importantly communication capability are the common characteristics of mobile devices. Recent advances in networking technologies such as 802.11 and Bluetooth enable these new mobile devices to connect to the Internet and local area networks.

In recent years, the power and capacity of these devices has become quite significant, and hence people are beginning to store and use their information across all these devices. For example, mobile phones which previously only stored contact lists are now used for storing pictures, music, video clips, calendars, task lists and notes.

As a result of the increased capacity of mobile devices, they are now used for storing much more information than before. Because the large amount of information stored on mobile devices, they need specialized information management functions. In the rest of this section, a taxonomy of mobility related issues for information management [13] is presented, as shown in Figure 1.

## 2. Management

### *a. Search*

Search capability in PIM, which lets users to retrieve information from different types of mobile devices, is crucial for accessing personal information.

There are significant differences between a search for personal information and a search in a huge amount of information collection like the Internet. Users generally have an idea about their personal information, they usually know some characteristics and context of the information that resides in mobile devices, because personal information is mostly created by the users themselves [5].

### *b. Power Issues*

Basically, power management can be defined as managing the battery's charge to optimize battery performance and longevity, preserving run and standby times even while adding more functionality to a mobile device. The concept has been around and it has kept importance since the birth of mobile devices because it can have a significant impact on the user experience.

Power management is critical because mobile systems are usually not tethered and they are dependent on the power source that comes with the system itself. Keeping a mobile device light and small requires a small battery. Consequently there is a serious need for an effective power management. Heat generated by the device is proportional to the amount of power consumed. Therefore, power management helps to reduce the heat generated on a mobile device which can affect the system and the environment in an adverse way. And probably the most important benefit of power management can be the prevention of disasters that could cause loss of user data on mobile devices [6]. A depleted battery can cause loss of critical information which is still in progress and in the volatile memory of the device. Power

management not only helps to sustain long battery life but it can provide appropriate methods to prevent unplanned shut downs.

Power management and information management on mobile devices usually intersect where PIM has auto procedures, such as indexing, information linking, etc. These automatic or periodic operations naturally consume power which is obviously scarce for mobile devices. Depending on the level of power available, the frequency of automatic indexing and linking operations may need to be adjusted to a lower cycle.

## 3. Content Capture & Production

### *a. Tagging*

Every bit of information that is stored on mobile devices should be associated with one or more relevant terms or tags that describe the item. These tags enable search and classification of personal information based on different categories.

Tagging allows users to attach their own attributes, typically one or more, to the information pieces created on mobile devices, this can be done manually by user by entering specific labels or automatically with predetermined classification information. Manual tagging could be done by assigning the appropriate information by the mobile user himself. Certain types of information such as system related information including date, time and device ID can be automatically associated. Depending on the abilities of mobile device, tagging information can be

broader and higher in details. For example, a mobile device with GPS receiver can create tags that contain geographic coordinates where the information is created.

There are mainly two techniques to achieve tagging Really Simple Syndication (RSS) and MPEG7. RSS presents a simple XML structure for describing metadata about content which is generally called “feed” [7]. A recent extension to RSS is the Atom syndication format [47]. MPEG7 (Multimedia Content Description Interface) is a standard for describing the multimedia content which lets users fast and efficient searching for material stored on mobile devices [8]. MPEG7 descriptions of content may include: title, copyright pointers, usage history, storage format, encoding, colors, objects and events, summaries, variations, user preferences and usage history. Of course, all these information are coded in an efficient way for searching and filtering purposes.

### *b. Production*

Today’s abundant and cheap storage makes it possible to record most life experiences in image, video, audio and other formats. Three ways of producing personal information are: manual creation, auto-capturing and synchronization. Calendar entries and to–do lists are mostly entered directly into mobile device database by its user as well as notes that contain simple text. Personal information in the form of still image, video and audio can be captured both automatically and manually. And in addition to all of the above, any type of personal information can be copied and created via synchronization with other devices.

### *c. Media type*

Captured content or produced material on a mobile device is stored in type of text, audio, video, image or structured documents. The content of audio could be music, ring tones or simply sound. Media type affects the size, content and tagging issues of the information kept on mobile devices.

## 4. Reception

By incorporating compact, inexpensive, self-activating sensors and using broader network capacities, mobile devices enable users capture information easily and passively [9].

Reception might happen in real time which may require the mobile device to expend more of its power. Alternatively, it can be done on demand or on a regular basis which can be scheduled before hand.

Conveying technique can use of SMS (Short Message Service), MMM (Multi Media Service), email or ordinary file format.

SMS can carry short text messages up to 160 characters in length when Latin alphabets are used and 70 characters in length when non-Latin alphabets, such as Arabic and Chinese, are used [14].

Besides the casual exchange of information among mobile users, the use of SMS has also expanded to other industries such as gaming, banking, education, remote sensor monitoring, advertising, voting, etc. [15].

MMS is a descendant of SMS. It extends text messaging to include longer text, graphics, photos, audio clips, video clips, or any combination of the above, within certain size limits.

MMS is frequently used to send photos and videos from mobile phones to other MMS capable devices or email accounts.

User context, device and network related issues also affect the reception.

For example, a mobile phone user might prefer only SMS when in a meeting. Device capabilities might limit the reception. A big picture or a picture with very high load of color information could not be displayed properly on a recipient device because of device characteristics.

### 5. Distribution & Transfer

Distribution and transfer have the same concerns as in reception over time and technique.

However repurposing has a significantly important role when distributing personal information. There are several hundred different mobile phones, each of which has different display, memory, processor and user interface characteristics. Networking profiles differ in bandwidth, latency, cost and availability in these mobile phones as well. It is obvious that to deliver information among various set of these devices need special handling which called content repurposing [10]. It primarily allows displaying and handling of different formats of information on numerous types of systems without need to change the underlying code.

Another important factor regarding distribution and transfer of personal information is format of the material. For all the improvements that have been accomplished in mobile world, there still are serious restrictions related to mobile devices compared to desktop computers. Small screens, limited network bandwidth and memory storage are among the limitations of mobile devices. Distributing every bit of information without cogitating the limitations of recipient mobile device will cause problems; like device failure, network congestion and needlessly expensive costs. The fairly easy solution to this problem is to send only the urgent and important part of the information in a form that device and network can manage and process [10].

## C. SEARCHING FOR PERSONAL INFORMATION

### 1. Overview

The number of ways to keep and manage information has increased considerably in recent years, thanks to the

overall increase in the diversity and the number of mobile devices, technologies, and applications on which users can rely. However, the accompanying fragmentation of personal information multiplies the chances of storing important information in the wrong place or form and forgetting it altogether.

Furthermore, users may encounter information which attracts their interests at the wrong time or at the wrong place. For example, we may stumble upon work-related information while we are at home and home-related information while we are at work. These unexpected situations usually result in save-immediately-search-later practices.

As a result, search is a crucial function in information management. Search helps to retrieve information from different types of mobile devices in order to efficiently access personal information.

Searching for personal information differs from other searches, especially search on the Internet. On the Internet, a user generally knows what the search result could be and where it could be. These and other properties can be helpful for implementing more effective search systems on mobile devices for the improvement of hunting personal information.

## 2. Background

Industry experts expect the demand for mobile search to grow because of the increasing appeal of mobile e-commerce. As the bandwidth of mobile networks become higher, they become more suitable for intense and complex activities; one

such activity is search. The resulting improved network performance will encourage users to use mobile search as it will be faster and cheaper [34].

Well known web search providers have begun providing new mobile search services. Most of these are adaptations of traditional keyword-based search for mobile use. These systems adapt the display of search results to fit the small screen of mobile devices.

## 3. Challenges for Mobile Search

Searching digital information through the use of mobile devices is impacted by displays on which only a small fraction of the set of ranked results can be displayed.

This makes for a very specific challenge: to find the best way of presenting search results on a mobile device. The easy but not very effective solution to this problem is to adapt the standard web strategy which involves displaying titles with URLs (Uniform Resource Locator) and informative snippets of the text. This approach brings its own set of problems, including the high demand for screen “real-estate” that is not available on mobile devices [39]. The elimination of informative snippets would create more harm than help in an attempt to remedy this problem, because they leave search at the mercy of vague result titles. It is obvious that, there is a need for a solution to present query results effectively on the small size of mobile device screens. Unfortunately, no miracle formula has been found so far. Some researchers suggest that the search effort can be reduced by user feedback indicating a single most relevant document in each display [32]. Jones et al. [40] also

address this problem in their research. Instead of using standard snippet text approaches, which involve the extraction of a block of document text, often related to the query, they use a group of key phrases, automatically extracted from result pages. The resulting key phrases provide for a more effective use of screen space and are at least as informative as using long result titles.

Existing mobile search engines still need improvements over coverage and relevance issues. Some queries either return empty results or produce misleading result lists which contain irrelevant results. To improve coverage, search engines ought to improve their skills to scrutinize the mobile resources. Since, mobile content tends to be short-lived and transient, traditional crawling techniques do not perform well on mobile internet. Moreover, mobile pages contain much less information than standard web pages which make them difficult for indexing purposes [39].

Many studies have highlighted some key observations about online queries. First, queries used on web searches tend to be short, generally 2 to 3 keywords on average, and as a result, a fewer number of terms has the potential to be non-specific and ambiguous. Second, searchers usually do not take advantage of advanced features, filtering options, and boolean operators to get better precision results. And finally, users often do not bother to look further than the first page of search results [34].

## 4. Related Work

The Google SMS service, launched in October 2004, enables mobile users to access the types of information most

likely needed by the user [35]. It is SMS based and allows users to search for information when mobile. In simple terms, the mobile user sends a query in a text message and receives the results in one or more SMS messages. The results in the SMS messages contain the information itself as opposed to the Google service, which provides the hyperlinks to other resources. The obvious advantages of using SMS is that it does not require the user to subscribe to special mobile data services; nor does it require upgrade to new devices since virtually all mobile phones are SMS enabled.

The Google SMS service is not aimed at personal information search, and it does not include any other mobile device apart from an SMS enabled mobile handset. It is mainly aimed at providing search for local business information, product prices, and similar specialized data.

Similar to Google SMS service, Leidner's wireless search engine [36] transmits results through SMS text messages, but it is based on natural language queries. In addition to traditional keyword based queries, his wireless search engine can accept natural language type questions and returns responses which contain likely answers rather than hyperlinks. The system actually still implements a desktop keyboard-based search, but the search results are mediated by a gateway between the mobile network and the Internet in an effort to overcome the output constraints. The gateway "repurposes" [42] the results of the desktop search for presentation on the small screen of mobile devices.

Raivio proposes [37] implementation guidelines for a Mobile Proxy, search functions and data modeling enabling an

optimal service solution which is located in the network and provides users a remote file storage space. Search functionality in a Mobile Peer-to-Peer (MP2P) system can be done in neighboring nodes, or can be extended to Edge and Super nodes, and other domains.

The search query can be broadcast to the adjacent edge nodes if the super node does not contain the information. In MP2P, there might be many nodes and any node can be any kind of mobile device. Each proximity consists of some number of mobile nodes which can exchange data between each other.

Another study [38] presents a general purpose distributed search service, for mobile file sharing applications and exchanging mobile documents, called Passive Distributed Indexing (PDI). It is constructed on peer-to-peer technology. It is asserted that it provides resource-effective searching for files residing on different mobile devices. The conveyance of queries and responses is done via a set of messages defined in PDI. As seen in P2P networks the forwarding of these messages is possible from node to node. A local broadcast transmission of query-and-response messages with the caching of results of popular queries at every device in PDI structure establishes the main building block of the system. With these building blocks, PDI gets rid of the need for flooding the whole network with search query messages.

The main design goal of PDI is to provide a general purpose search service in mixed application domains and system environments; whereas we focus on personal information search scattered across several different devices including mobile devices.

## D. SUMMARY

Section B offered a conceptual framework detailing the taxonomy of PIM, as an activity consisting of four sub-activities: management, content capture & production, reception, and distribution & transfer.

Section C defined the search function in PIM which enables the user to find and retrieve a collection of personal information. A survey of existing search tools was also provided, along with a discussion of challenges for mobile search. Related work about mobile search also explored in this section.

THIS PAGE INTENTIONALLY LEFT BLANK

# III. MOBILE DEVICES AND INTERCONNECTIVITY

## A. OVERVIEW

Mobile devices such as PDAs, mobile/smart phones, and mobile computers are becoming increasingly ubiquitous and transforming everyday lives both at home and in the office. They profoundly diminish the distinctions between communication and media creation, in-person interaction and telecommunication, and real and virtual environments.

Additionally, because of recent advances in wireless networking technologies such as Wi-Fi, mobile data services and Bluetooth, these new mobile devices have potential uses that greatly surpass previous intentions.

## B. MOBILE DEVICE CATEGORIES

### 1. Mobile Phones

A mobile phone is essentially a micro-computer, which is battery-powered, and contains one or more wireless transmitters and receivers optimized for voice input and output. Even the simplest model has a keypad, an LCD display, and a general purpose computing platform, typically Java Mobile Edition or .NET Compact Framework. More advanced models come with an integrated camera, a few gigabytes of local storage, and multiple wireless interfaces including Bluetooth and even Wi-Fi [18].

In addition to mobile communication, mobile phones support an array of functions ranging from that of a simple digital organizer to that of a low-end personal computer.

Since they are designed for mobility, they are lightweight and compact enough to carry in a pocket.

Despite the numerous different models found on the market, most mobile phones have a group of comparable features and capabilities. They have a microprocessor, read only memory (ROM), random access memory (RAM), a communication module, a digital signal processor, a microphone, a speaker, a keypad, and a display. A mobile phone also has an operating system (OS) which resides on ROM along with some system and registry files. RAM, which is generally used to store a user’s information, is kept alive by battery power. When the battery fails, the information in the RAM can be lost [19].

Modern mobile phones are generally equipped with system level microprocessors, which cut down on the number of supporting chips, and they help to decrease the overall power requirements of devices. Built-in support of memory expansion cards, and interfaces for additional specialized peripherals are among the features of latest mobile phones. Wireless communication interfaces may also be found on mobile phones including Bluetooth, Wi-Fi or infrared.

The line that separates mobile device categories has been blurring recently. Some mobile phones include other devices such as GPSs (Global Positioning System) and media players.

Independent of the type of a mobile phone, virtually all mobile phones offer text and voice messaging services, and fundamental PIM applications including an address book and a calendar. Most of the mobile phones also provide means to synchronize personal information with other mobile

devices and personal computers. More improved models support multimedia messaging, enable direct browsing of the Internet with built in web browsers, let users exchange emails, and instant messaging.

### 2. PDA

PDAs (Personal Digital Assistant) are highly portable and personal computing appliances, which can be carried around and used anytime and anywhere. PDAs can be used for a variety of functions: calculation, a clock and calendar, accessing the Internet, sending and receiving emails, video recording and storing data, typewriting and word processing. They can be used as an address book, used to make and write on spreadsheets, used as a radio or stereo, as well as for playing computer games, recording survey responses, and GPS (Global Positioning System) receivers. Newer PDAs also have both color screens and audio capabilities, enabling them to be used as web browsers, or portable media players. Many PDAs can access the Internet, intranets or extranets via Wi-Fi, or WWANs (Wireless Wide-Area Networks). One of the most significant PDA characteristics is the presence of a touch screen [21].

Modern PDAs are very small in size, fit comfortably in a pocket and generally have good battery life, which can be recharged at night or when at the office. A PDA usually includes a small screen, which is often bigger than a digital phone but smaller than the smallest notebook computer, and a small QWERTY keyboard that is made for thumb typing and a stylus which is a metal or plastic pen to input data or communicate with the device by a touch pad screen. A

PDA might also include handwriting recognition software, voice recognition, and a digital voice recorder.

While components and specifications differ among models, recent PDAs have more in common. They come equipped with lots of RAM, storage in either miniature hard drives or flash memory cards or sticks. Newer models have USB interfaces which support a variety of peripherals to use with PDAs. Some models come with a suite of software programs preinstalled, while others offer optional programs if desired.

A PDA might also incorporate mobile phone functionality and wireless local area network (LAN) capability. It can connect to the Internet in order to check email, send messages, or even monitor the stock market. With flash card capability, a PDA can store, access, and transfer virtually any kind of data, including maps, spreadsheets, presentations, and dockets.

Some experts insist that the rising popularity of smartphones means an end for PDAs, relying on the argument that consumers want one device that does it all. While there are many PDAs in current use, the trend shows that PDAs are merging with cell phones. In the near future, this might eliminate PDAs as a distinct class of devices [22].

### 3. Smartphones

Technological improvements in the mobile phone market have created a new type of mobile device called smartphone. The most significant features of a smart phone include Internet access, e-mail access, scheduling software, built-in camera, contact management, and the power to run a wide

variety of general and special-purpose applications as well as occasionally the ability to read business documents in a variety of formats such as reports, slides and spreadsheet files. Some smartphones add extra features such as touch screen displays and tethered modem capabilities on top of the default phone characteristics. A rich email support is an indispensable and a characteristic key feature found in a smartphone [20].

Mobile phones whether basic or advanced typically use a vendor proprietary operating system. Smartphones generally run on one of the following operating systems: Palm OS, Windows Mobile (phone edition), RIM OS, Symbian OS, or Linux. These operating systems support multi-tasking and they are designed to match the capabilities of smartphones. They often provide Java Virtual Machine support or native application support using a SDK (software development kit) for a programming language [19].

As innovations allow mobile handheld devices to add more functions to their feature sets, the difference between these two gadgets becomes less clear. The differences between mobile phones, smart phones and PDAs are somewhat blurred. This has complicated efforts for reaching commonly accepted definitions.

**4. PC/Notebook**

Given that wireless connectivity is available everywhere, laptop computers are being used as truly mobile devices. In order to describe a laptop/notebook as a mobile device, it must have some standard features. A mobile computer must be light enough to carry all around, it must

do all of the things which can be done with a desktop computer, and it should be able to use the same software as its counterpart, the desktop computer [23].

Notebook computers, or simply notebooks, normally run on a single battery or from an external power supply which also charges the battery while supplying power. Notebooks contain similar components that are found in desktop computers. They characteristically have LCD screens and built-in keyboards, and many of them are equipped with a touchpad which is an input device that enables one to move the cursor through finger motions. In addition, an external mouse can be attached. The components of a notebook computer generally are reduced in size and optimized for mobile use and efficient power consumption.

Integrated modems and network adapters, standard serial and parallel ports on a notebook computer make it easier to work on mobile when away from office or home. Wi-Fi network adapters make notebooks as easy to use with peripherals as a desktop computer and help sustain mobility.

Since there are no universal standards for notebook computer design and parts, it becomes very difficult and costly to upgrade their basic components. Furthermore, to save space and cost, manufacturers generally produce notebooks with many of the standard elements already integrated on the motherboard. Some exceptions to this include RAM modules, hard drives and batteries. These issues hinder the upgradeability of notebook computers, thus creating higher costs in the long run.

Despite the limitations as compared to the desktop computers, notebooks remain the preferred choice of mobile

users who require data intensive applications. PDAs and smart phones provide convenient access to corporate information such as email and personal information, but these devices can not compete with notebooks because of their tiny screens, convoluted user interfaces and restricted keypads that prevent heavy traffic from power users. In spite of the recent improvements on other mobile devices, notebooks will still be the most important mobile device for remote users. Notebooks have advanced in their power and battery utilization as well as their size and weight. Finally, by embedding mobile broadband, manufacturers make notebook computers truly mobile which are no longer bounded by fixed-line connections such as Ethernet or even by Wi-Fi hotspots, which still tether users to a location [24].

### 5. Handheld Computers

Handheld computers constitute another subcategory of mobile devices. A handheld computer often fits in a pocket and it comes with a tiny keyboard for user input, a relatively large display for user output. These mobile devices are usually manufactured in a clamshell-like package with a rich set of connectors [6]. The main difference between PDAs and handheld computers is that the handhelds are usually equipped with a miniature keyboard, unlike PDAs' dual purpose (keyboard and display) touch-screen interfaces. Handhelds are used to achieve a variety of tasks for increasing efficiency that include digital recording, storing notes and documents, sending and receiving invoices, information management, and scanning barcodes.

## C. CONNECTION SCHEMES

### 1. Personal Area Networking

#### *a. Bluetooth*

Bluetooth provides a way to connect and exchange information between devices such as mobile phones, PDAs, PC/Notebooks, printers, digital cameras, and video game consoles over an unlicensed short-range radio frequency.

The key features of Bluetooth wireless technology are robustness, low power usage, and low cost. Bluetooth offers services that enable the connection of devices and the exchange of a variety of data between these devices [16].

With the help of Bluetooth technology, many cables that connect one device to another can be replaced with one universal short-range radio link. For example, a mobile phone equipped with Bluetooth radio technology and a notebook would replace the burdensome cables used before to connect a mobile phone to a notebook computer. PDAs, desktop computers, keyboards, headsets, and almost any other digital device can be a part of Bluetooth network system [17].

Bluetooth enables a mechanism to construct small private ad hoc groupings of mobile devices away from fixed network infrastructures. It is very resilient to noise which makes it easy to operate in a noisy radio frequency environment such as a home or an office. The Bluetooth radio system runs on a frequency-hopping scheme and uses a fast acknowledgement design to make the connection quality robust.

Bluetooth can reach a maximum data capacity of 1 Mbps, which is equivalent of only 780 Kbps when the protocol overhead is taken into account.

Most of the new mobile phones are manufactured to provide Bluetooth connection; some other mobile devices such as PDAs and notebooks can have either integrated Bluetooth radio modules or optional add-on devices which support the Bluetooth radio system.

Bluetooth can be used on mobile phones in several ways. It provides cable-free remote networking with another mobile phone, PDA or a notebook computer. It enables mobile phone personal information synchronization with trusted mobile devices. And, it provides wireless hands-free operation using a Bluetooth headset. Of course there might be other usage models which are not discussed here.

***b. Infrared***

Infrared data connection is generally called with the name IrDA (Infrared Data Association). IrDA actually defines a standard for an interoperable universal two way cordless infrared light transmission data port for uses such as personal area networks (PANs). IrDA is utilized for high speed short range, line of sight, point-to-point cordless data transfer - suitable for mobile phones, digital cameras, handheld data collection devices, etc. [27].

The data standard of IrDA had found a place in millions of notebook computers, mobile phones, PDAs, and other devices until Bluetooth connection interface has became very popular. The market coverage of infrared interface soared swiftly because infrared transmissions are

inherently localized and governments do not regulate the infrared portion of the light spectrum. It helps, too, that the components of an infrared transceiver have been perfected and their costs reduced through their successful use in so many remote control applications.

The first IrDA standard paved the way for asynchronous data communications at rates up to 115.2Kbps and synchronous communications at 1.152Mbps and a synchronous 4Mbps option was added to the standard later.

IrDA transceivers communicate via infrared pulses in a cone that extends a minimum of 15 degrees half angle off center. The position and location of devices have critical importance during infrared connection. The IrDA physical specifications require that a minimum irradiance be maintained so that a signal is visible up to three feet away. Likewise, the specifications necessitate that a maximum distance not be exceeded so that a receiver is not overwhelmed with brightness when a device comes close. It is obvious that in order to use infrared connection there must be a line-of-sight visibility between device infrared ports.

It has been argued that Bluetooth has been created as a substitute for infrared. However this is not quite true: Bluetooth has been invented as an enhancement to Infrared, especially in terms of low cost, small volume, low power, and the presence of infrared's limitations, such as the its unidirectional connections, its short connectivity range of only a few feet and its limitation to point-to-point connections. Although Bluetooth has actually coped with these constraints with a connectivity range of up to 30 feet, or its capability for point-to-multipoint connections,

the two technologies are quite complementary. While Bluetooth is very applicative for mobile networking, infrared is more appropriate for the direct connections, e.g. for exchanging small files [28].

Recent notebook computers and mobile phones do not have an infrared port mostly because of in favor of Bluetooth. Many experts claim that infrared connection technology on mobile devices has come to an end. In his article [29], Brooks wrote “Many are calling IRDA the most popular failure in mobile technology—nearly every mobile computer or device carries an IR port, and yet those ports are severely underused.”

### 2. Mobile Data Services

Mobile Data Service (MDS), which carries digitized information using wireless technology and personal communication systems, supports a vast array of services over telecommunications networks, including the Internet [18]. A few of such services include short messaging service (SMS), multimedia message service (MMS), email, download services, and Internet connection services e.g. General Packet Radio Service (GPRS).

Although there are many mobile wireless technologies, there are just two major technology families deployed in the mobile market: Global System for Mobile Communications (GSM) and Code Division Multiple Access (CDMA). In both networks, mobile users have access to voice and data services. Both in GSM and CDMA, data access is over a channelized medium, where separate wireless frequency channels or timeslots are dedicated to data communication and signaling [31]. GPRS is

the world's most ubiquitous wireless data service, available now with almost every GSM network. It can provide rates of up to 115 kbit/s [41].

A data communication process between a mobile phone and an Internet host simply happens as follows. First, a mobile phone that wants to send and receive IP packets starts by requesting a packet data protocol context from the service provider [18]. The service provider sends back an IP address, a packet data protocol, a quality of service specification and, optionally, a DNS name. By way of this process, a mobile service provider can associate the mobile phone’s unique identification number (IMEI) with its IP address. From that point on, a standard mobile phone location system guarantees that any data packet sent to the mobile phone’s assigned IP address be routed to an actual device wherever it roams. At this point, a mobile phone is no different than any other Internet host, and it can exchange IP packets with any other device that have a connection to the Internet. It should be noted that, the connection is initiated by a mobile phone in the process. If an Internet host tries to connect to a mobile phone, it would not be so easy [22] and that is because every time a mobile phone establishes a GPRS connection, it is provided with a different IP number that is exclusively accessible within the service provider network. Also these addresses are routable within the network only, which means that only the users of the same service provider can reach each other. Unless the service provider assigns a global IP number and provides NAT (Network Address Translation), the mobile phone is not accessible from the Internet.

Data services accessed through mobile phones are still rare. However, shrinking hardware and service costs and subsidized handsets are enabling the mobile phone market to grow. Additionally, heavy competition among service providers causes voice service margins to shrink, so they have to find ways to keep existing relationships and huge cash flows. One way of this is to fund innovation and provide data services and seamless worldwide roaming. These factors are expected to make data services on cell phones rapidly gain popularity and diversity [30].

### 3. WLANs

Wi-Fi is a wireless technology brand owned by the Wi-Fi Alliance to describe wireless local area network (WLAN) products that are based on the Institute of Electrical and Electronics Engineers' (IEEE) 802.11 standards. A Wi-Fi enabled device such as a notebook, mobile phone or PDA can connect to the Internet when within range of a wireless network connected to the Internet. The area covered by one or several interconnected access points is called a hotspot. Hotspots can cover areas as small as a room with wireless-opaque walls or as large as many square miles covered by overlapping access points.

Mobile users do not necessarily need to use wireless interfaces; a mobile user can simply connect to fixed networks using wired adapters after he changes location. Consequently, mobile systems and wireless systems do not mean the same thing. It must be stated that they converge at many points, though. While mobile networks help to keep

track of the location and support for routing, wireless networks take care of bandwidth allocation and error checking issues [25].

Wireless networks usually provide coverage in a small area, such as a building, a park, or a hallway by spreading or replacing wired networks. The mobility and flexibility of wireless LANs generally surmount the bandwidth concerns. Wireless LAN users have to share the bandwidth which sometimes creates high traffic and slow connections, whereas mobile networks are different, in this sense, by allocating a separate channel for every user [25].

Wi-Fi also supports direct connection, also known as ad-hoc connection, between mobile devices without the existence of a fixed access point and where all nodes in the network behave as routers and take part in the discovery and maintenance of routes to other nodes in the network.

A network containing tens to thousands of mobile devices connected with each other in such an ad hoc fashion is called as Mobile Ad Hoc Network (MANET). In this type of network, all nodes can freely and arbitrarily move in any direction with any velocity, and routing takes place without the existence of fixed infrastructure [26].

It must be emphasized that not only mobile devices can use Wi-Fi, but many consumer devices such as desktop computers, printers, and digital cameras might also benefit from wireless LANS.

# IV. SYSTEM ARCHITECTURE AND IMPLEMENTATION

## A. OVERVIEW

The more things can be saved on mobile devices, the more it becomes difficult to find the information one is seeking. It was easy when only phones were limited in their capability but as the devices have become more resourceful, their usage has expanded enormously. Today’s mobile phones are comparable in compute capability and storage to the desktop computers of a decade ago. Consequently, searching and reaching the desired content on a mobile phone has been become an important function. Users and businesses are more dependent on mobile devices than ever before. Managing information scattered across multiple devices is becoming a growing problem. This chapter describes the implementation of a search system for efficiently and conveniently searching through personal information stored on multiple mobile devices.

## B. SEARCH

### 1. Aim

The aim of this system was to test the feasibility of providing search ability as a part of personal information management for mobility. Observations were also made which related to the practicality of such an implementation.

## 2. Central Idea

Considering that PIM should be expanded to include functions that enable the users to access urgent information whenever and wherever possible [10], the system is designed in a way that lets the user use any available mobile device within reach to accomplish the required search.

In other words, the user can initiate search from any of the four device types, depending on his choice; these are the mobile phone, PDA, the personal computer and the Internet. Actually, for this application, the word internet means any computer connected to the Internet. Since most people keep at least some of their personal information on Internet (email account, file server, etc.), expanding the search to the online data is necessary.

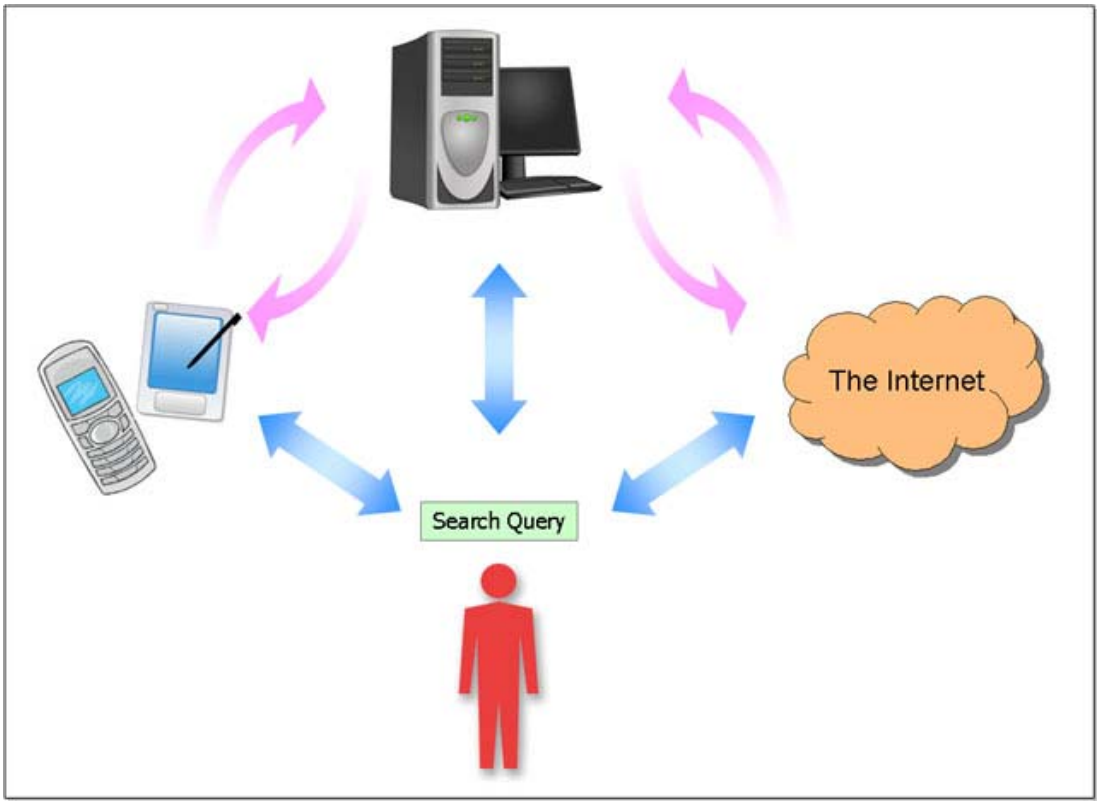


Figure 2. General Architecture.

The main reason for including all four platforms into this search system is to provide flexibility and availability to the users all of the time. If the user is able to access his personal computer which has a large screen, he should be able to use it to search for his personal information as well as a small screen mobile phone.

When a person runs into a friend on a busy street, most probably he will store the friend’s new phone number on his mobile phone. Similarly after receiving an important call from a new number for the first time, it is so easy to store that number along with that contact’s name into the mobile phone address book. Needless to say, it is so burdensome to copy these numbers into the address book on one’s personal computer. As a result, while phone numbers can be stored in several places including email address books, often they are primarily stored in the cell phones. In some enterprise environments, employees’ phone numbers are available in email contacts but it is a small subset of all of the phone numbers most people need to maintain.

One of the most attractive and relatively trouble-free places for a user to save personal information is the Internet. These days multiple gigabytes of storage are provided, without charge, to users for archiving email messages, notes and contact information. And, this storage is accessible from anywhere in the world provided that an internet connection is available. Due to its popularity, it is also needed to include the online data in this search system. Hence, any useful and convenient search system for personal information must cover mobile phones, PDAs, PCs and the personal data residing on the Internet.

## 3. High-Level System Architecture

The search system gives the user the freedom to initiate search from his or her preferred available device. If a mobile phone is the only available device, which happens frequently for a mobile user, search can be initiated from the mobile phone. Sometimes an internet café is the only way for a user to reach valuable personal information stored in the personal computer at home.

In our system, by default, search is first performed on the device on which the query is issued. The rationale is that if the information is available locally, it can be quickly retrieved and delivered. The user can, however, expand the search to include other devices as seen in figure 3.

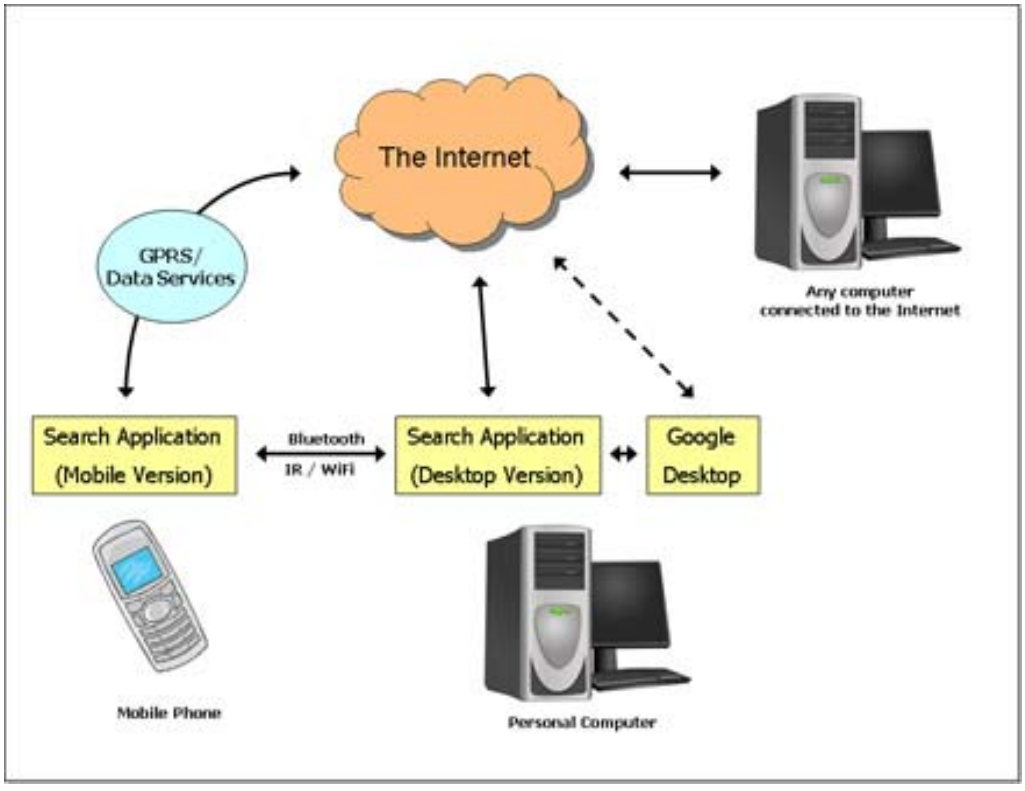


Figure 3. High Level Architectural Diagram.

For example, the user is using looking for a friend's contact information and she has the mobile phone with her, when she presses the search command on her mobile phone after entering the search criteria, the software on the phone first searches the phone's local database. If the information is found, it simply displays the information and the search process ends.

If the user prefers to extend the search, three more options are available to make the search to include PDA, the personal computer, or as another option extending it to the Internet. A sample screen is seen on figure 4. Assuming that, all options are included, if it can not find the required information in the phone local database, the phone sends the search query along with related criteria to the personal computer.

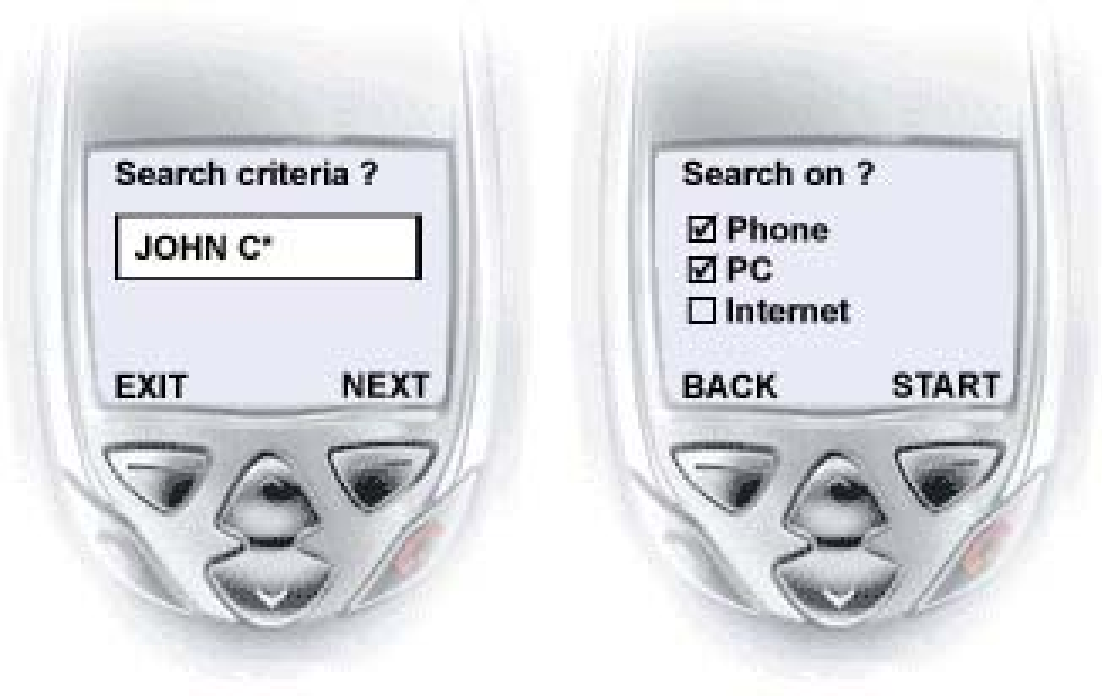


Figure 4. Starting a search on the mobile phone.

After the system running on the PC receives the query, it begins to search the PC’s local database and if it finds the requested information, the PC responds to the cell phone with the search results. If it can not find the information, the system extends the search to Internet. Results are finally aggregated and sent to the mobile phone.

## 4. Design, Architecture and Implementation

Because it is the most supported and accepted environment for mobile phones, Java ME (Java Platform, Micro Edition) [43] was used to implement the software running on the mobile phone. Mobile phones typically have an address book (or a contact list) to keep track of people one would like to stay in touch with, a calendar to keep track of important events, and a to-do list to keep track of items one does not want to forget. Next to placing a voice call, accessing this type of personal information is possibly the most important function in a mobile phone. The PIM API for Java ME [45] supports mobile Java applications to read from and write to the locally stored databases of personal information. Even though, protected system and OS files are generally beyond the direct reach of any user application, files crated or copied by the user are accessible and thus searchable by the user. This can be done with the help of another optional package for Java ME the FileConnection API (JSR-75) [45] that gives access to the mobile phone's file system including removable storage media like memory cards.

The PDA and desktop portion of the software was also developed in Java as well; many other programming languages, such as C#, C++, etc. are also available for these platforms though.

As is always necessary, when working with mobile devices, due to the changing mobile device capabilities and functions (power level, network coverage, etc.), users should be able to select how the search function be extended, and which platforms will be included. This will also limit the time spent when a quick-local search is needed.

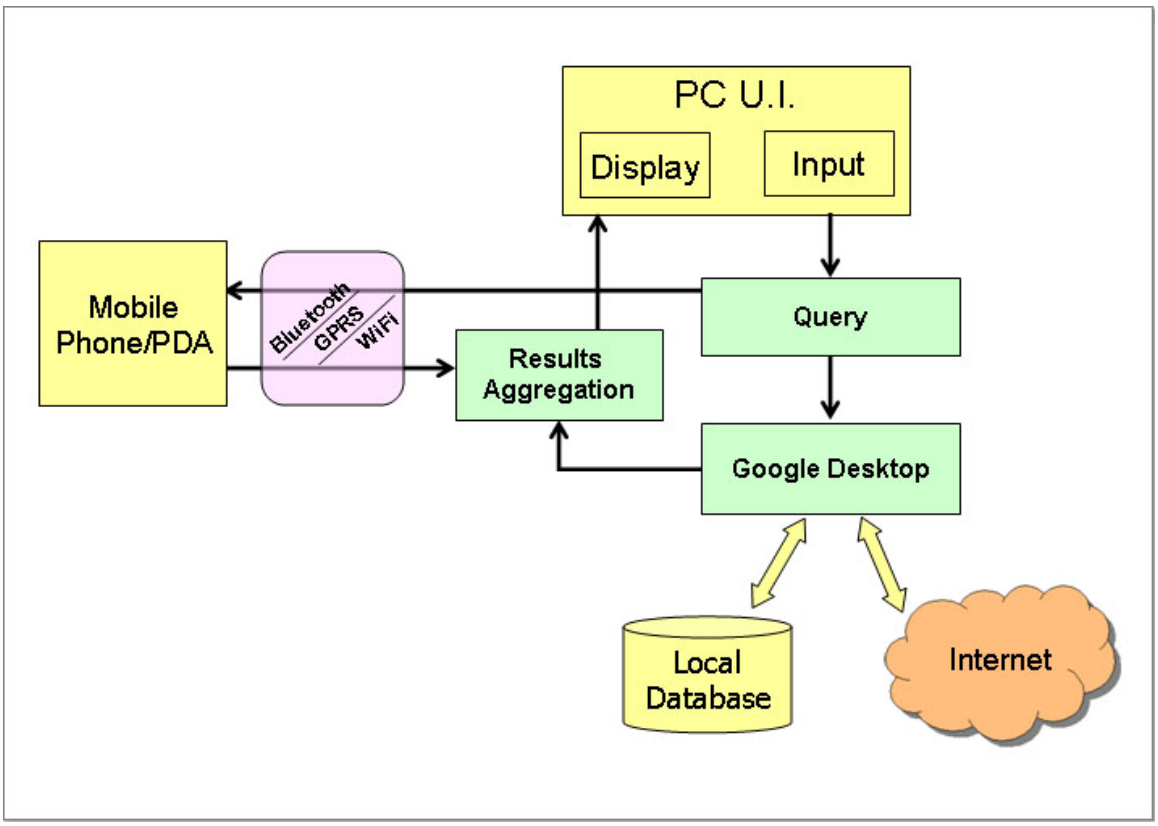


Figure 5. Component-Level Diagram.

In this search system, Google Desktop [44] is used for computer and web searching. This tool expands the search process further to the Internet, if it is asked to do so. It is actually a desktop search application which provides full text search over emails in addition to the files saved on the computer.

Searching on the personal computer, and exchanging search queries from and results to the mobile phone is maintained by a component of the system running on the desktop computer. This application was created in Java and integrated with Google Desktop. The main feature of this program is to enable search function when the user starts search on the personal computer. The integration between the application and Google Desktop is done with the help of GDS Java API [46]. It is a simple interface for access to Google desktop search from any java application. A component level architectural diagram is presented in figure 5.

## C. OBSERVATIONS

The communication between the mobile phone and the PC is currently accomplished with Bluetooth. The current mobile phones generally come equipped with a Bluetooth class-2 capability which has a communication range of approximately 10 meters [16]. Bluetooth lets these devices communicate with each other when they are in range. Since it uses a radio communications system, the devices do not have to be in line of sight of each other, or even in the same room, as long as the received transmission is powerful enough.

More Wi-Fi enabled mobile phones have begun appearing on the market. Future work in this search system could be to provide the option of using Wi-Fi connection between a mobile handset and personal computer.

GPRS (General Packet Radio Service) is the world's most ubiquitous wireless data service, available now with almost every GSM network. A GPRS connection between a mobile phone and a PC has its own consequences and hurdles. During the

experimentation, it was seen that every time a mobile phone establishes a GPRS connection, it is provided a different IP number that is exclusively accessible within the service provider network. These addresses are routable within the network only, which means that only the users of the same service provider can reach each other. Unless the service provider assigns a global IP number and provides NAT (Network Address Translation), the mobile phone is not accessible from the Internet. But the other way around, that is accessing a PC from a mobile phone via GPRS, is possible and it has been tested and the search system has been already been tried over such a connection. One way to overcome this problem might be to establish the GPRS connection from mobile phone to PC even when the search request has been initiated from the PC.

## D. SUMMARY

In transitioning from 1G to 3G, mobile phones have become more and more powerful. Today's mobile phones are capable of storing not only phone numbers but also more detailed personal information as well as traditional computer files. Their enhanced capability lets users store and carry more information on their mobile phones, which increases the importance of search on these devices. In addition, people continue to use their "traditional" devices such as PDAs, PCs and hosted servers.

With today's technology it is possible to reach the entire Internet from a mobile phone, but with its limited bandwidth and small display, finding and retrieving the desired information can be really tedious. With this in mind, a simple but effective context sensitive personal

information search system was implemented and presented in this chapter which covers the entire range of mobile devices a user may use.

What makes this system different from many other mobile search systems is that the search for personal information can be begun from any of the user’s devices: mobile phone, PDA, the personal computer or the Internet. Whichever available device is chosen, the process lets the user to extend the search over to the other platforms. After search has been completed the results are brought to the user on whichever device first started the search process.

# V. CONCLUSIONS AND RECOMMENDATIONS

## A. CONCLUSIONS

This thesis has been aimed at improving the knowledge base for search on mobile devices regarding personal information. Today's mobile users encounter a wide range of problems in managing personal information, and consequently there is a need to develop improved interfaces to better support this everyday activity. This research focused on one specific area of PIM (Personal Information Management), that of searching information scattered across multiple devices. As discussed in Chapter-II, previous research and applications relating to this area have been limited. Although many studies of "search on mobility" have been carried out, few have considered user needs beyond the boundaries of specific environments. There is a lack of applications for multi-platform search on mobile devices aimed at improving PIM. Most research has been focused on the display of search results to fit the small screen of mobile devices. Also, there are physical limitations, such as sluggish network speed and slow processor performance, which have limited mobile applications from advancing more rapidly. Users can thus wait a long time, particularly when wireless signals is weak, for query responses. In addition to being inconvenient, this can be time-consuming and, therefore, expensive for mobile customers.

A demonstration search system was developed to validate the feasibility of implementing a multi platform personal information search on mobile devices. It was discovered that

with today’s technology it is possible to manage personal information from a mobile phone, but with its limited bandwidth and small display, finding and retrieving the desired information can sometimes be tedious. And, without service providers’ active involvement in removing cross-network barriers, an ideal implementation of remote managing is very difficult.

## B. LESSONS LEARNED

Mobile search is likely to change and improve when more advanced devices and services become more wide-spread, such as faster 3G-network technologies. Increased speeds improve browser-based searching and, more importantly, enhance advanced applications and encourage more users to try services other than telephony. Better PIM tools and applications can make users get more value out of their mobile devices.

While several desktop search applications provide access to personal information, it is evident that a broader approach is needed to manage different types of devices that a user can have. Simple keyword search capabilities should be improved with user interfaces to allow users to define their information needs based on various cues, and to view and refine results quickly and flexibly.

## C. FUTURE RESEARCH

Mobile and wireless networks are the next wave of networking because they are truly help the mobile workforce in an increasingly information centric society [25]. But, there have been many challenges which relate to hardware,

software and design issues. There are a number of opportunities for improvement which stem from mobile users' needs which mostly involve with personal information management. One of the motivations for this thesis has been to benchmark the early stages of mobile search with a view to emphasizing the importance in information management.

The mobile Internet is becoming the primary form of anytime-anywhere information access. Naturally, the appearance of next-generation mobile devices and networks will help to advance the quality of mobile services, and thus user experiences. There are positive results [34] obtained after research in this area which suggests that the role of mobile search is on the rise and with many of their powers big players are now looking to the mobile sector as a new frontier.

Significant usage impact has already been made by the early adopters of mobile search but these applications continue to struggle with the peculiar challenges presented by the mobile world, especially in relation to limitations of display and text input.

The search system presented in Chapter-IV can be improved or extended in several ways. The first improvement is in the connection between the PC and the mobile phone. When the search is started on a device other than the mobile phone, the search could not be extended to include mobile phone unless it is in the range of Bluetooth connection. The search application on the mobile phone has the ability to listen Bluetooth connection requests, which also enables the conveying of queries and search results. But in a situation where the mobile phone is far from the other devices

(personal computer, a remote internet connected computer, etc.), there is no way to trigger search application on the mobile phone. The main cause of this problem, as discussed in Chapter-IV, Observations section, is the cell phone routing of traffic which is limited to within the cellular service provider network. A bridge needs to be built between the two networks.

Ideally, a field study should have been conducted after the prototype was implemented to identify users' needs in the mobile context by observing usage of the search system in realistic settings. Due to limited time, this study could not be done. It will be useful to conduct this study to check how well our system meets the user requirements.

# APPENDIX. DEMO APPLICATIONS

## A. MOBILE PHONE

### 1. OVERVIEW

The search application which runs on mobile phone is written in Java ME [43]. This application, searches the mobile phone and displays the results if found, otherwise it sends the query to personal computer via Bluetooth. Mobile phones typically have an address book (or a contact list) to keep track of people one would like to stay in touch with, a calendar to keep track of important events, and a to-do list to keep track of items one does not want to forget. Next to placing a voice call, accessing this type of personal information is possibly the most important function in a handset. The PIM API for Java ME [45] lets mobile Java applications read from and write to the locally stored databases of personal information. Even though, protected system and OS files are generally beyond the direct reach of any user application, files crated or copied by the user are accessible and thus searchable by the user. This can be done with the help of another optional package for Java ME the FileConnection API [45] that gives access to the mobile phone's file system including removable storage media like memory cards.

In the next section, the Java code for important sections of this application is listed.

## 2. PROGRAM CODE

```
/*
 * Created on Nov 16, 2006
 */
package mobilephone.mobcon;

import java.io.IOException;
import java.util.Vector;

import javax.bluetooth.ServiceRecord;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.Gauge;
import javax.microedition.lcdui.Ticker;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;
import javax.microedition.rms.RecordStoreException;

import mobilephone.btcom.connection.IConnectionListener;
import mobilephone.btcom.connection.SerialConnection;
import mobilephone.btcom.device.ISelectBTConnectionListener;
import mobilephone.btcom.device.SelectBTConnectionScreen;
import mobilephone.btcom.rms.SettingStorage;
import mobilephone.btcom.tools.ILogger;

import java.io.EOFException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.Enumeration;

import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;
import javax.microedition.lcdui.Choice;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Item;
import javax.microedition.lcdui.ItemStateListener;
import javax.microedition.lcdui.List;
import javax.microedition.lcdui.Screen;
import javax.microedition.lcdui.TextBox;
import javax.microedition.lcdui.TextField;
import javax.microedition.rms.RecordComparator;
import javax.microedition.rms.RecordEnumeration;
import javax.microedition.rms.RecordFilter;
import javax.microedition.rms.RecordStore;

/**
 * Midlet to search contact on PC.
 *
 * @author Mehmet AKBAS
 * @version $Id: v 1.5
 */
public class MobileControlMidlet extends MIDlet implements
        ISelectBTConnectionListener, IConnectionListener, CommandListener {
```

```
    public String cevap;

    // STATUS OF MIDLET
    private final int STATUS_INITIAL = 0;
    private final int STATUS_READ_SETTINGS = 1;
    private final int STATUS_SEARCH_URL = 2;
    private final int STATUS_WRITE_SETTINGS = 3;
    private final int STATUS_OPEN_SERIAL_CONNECTION = 4;
    private final int STATUS_CLOSE_SERIAL_CONNECTION = 5;
    private final int STATUS_CLOSE_APPLICATION = 6;
    private final int STATUS_REQUEST_APPS = 7;
    private final int STATUS_SELECT_APP = 8;
    private final int STATUS_REQUEST_CONFIG = 9;
    private final int STATUS_CONTROLLING = 10;
    private final int STATUS_SEND_SHUTDOWN = 11;

    private final int STATUS_SEARCH = 12;
    private final int STATUS_DISPLAY_SEARCH = 15;
    private final int STATUS_BROWSE = 13;
    private final int STATUS_ADD_NEW = 14;

    private int status = STATUS_INITIAL;
    // ADRESSBOOK - BEGIN
    private RecordStore addrBook;
    private static final int FN_LEN = 10;
    private static final int LN_LEN = 20;
    private static final int PN_LEN = 15;
    final private static int ERROR = 0;
    final private static int INFO = 1;
    private Display display;
    private Alert alert;
    private List mainScr;
    private String[] mainScrChoices = { "Search", "Add New", "Browse"};
    private Form searchScr;
    private TextField s_lastName;
    private TextField s_firstName;
    private Form entryScr;
    private TextField e_lastName;
    private TextField e_firstName;
    private TextField e_phoneNum;
    private List nameScr;
    private Vector phoneNums;
    private Form optionScr;
    private ChoiceGroup sortChoice;
    private TextBox dialScr;
    private int sortOrder = 1;

    private final Command cmdAdd = new Command("Add", Command.OK, 1);
    //private Command cmdBack;
    private final Command cmdCancel = new Command("Cancel", Command.BACK, 2);
    private final Command cmdDial = new Command("Dial", Command.OK, 1);
    //private Command cmdExit;
    private final Command cmdSelect = new Command("Select", Command.OK, 1);
    private final Command cmdSearchNetwork = new Command("Computer",
Command.SCREEN, 4);
    private final Command cmdSearchLocal = new Command("Phone", Command.SCREEN,
3);
    private final Command cmdSearch = new Command("Start", Command.SCREEN, 3);
    // ADRESSBOOK - END

    // COMMANDS
    // return from list form
```

```
    private final Command cmdBack = new Command("Back", Command.BACK, 0);
    //  break a progress bar
    private final Command cmdBreak = new Command("break", Command.BACK, 0);
    private final Command cmdExit = new Command("Exit", Command.EXIT, 0);
    private final Command cmdAbout = new Command("About", Command.SCREEN, 0);

    // DISPLAYS
    private final MobileControlList listDisplay = new MobileControlList(null,
            new Command[] { cmdExit, cmdBack, cmdAbout }, this);

    // screen, which cares for device search and url selection
    private final SelectBTConnectionScreen connectScreen = new
SelectBTConnectionScreen();

    // progress bar used in anonymic forms.
    private final Gauge progressbar = new Gauge("", false, Gauge.INDEFINITE,
            Gauge.CONTINUOUS_RUNNING);

    // Gauge Form with one command only for progress bar
    private Form gaugeForm = new Form("wait") {
        private Command lastCommand;

        /**
         * add command and remove existing ones.
         *
         * @param cmdToAdd : the Command to add
         * @...lcdui.Displayable#addCommand(javax.microedition.lcdui.Command)
         */
        public void addCommand(Command cmdToAdd) {
           //remove old command
            if (lastCommand != null) {
                this.removeCommand(lastCommand);
            }
            lastCommand = cmdToAdd;
            super.addCommand(cmdToAdd);
        }
    };

    // CONNECTIONS
    private String url = "";
    private String deviceName = "";
    private boolean connected = false;
    private SerialConnection connection;

    // implement a NULL-logger
    private ILogger nullLogger = new ILogger() {
        public void log(final String str) {
            // do nothing
        }
    };

    // APPLICATION
    private String currentApplication;
    private static final char DELIMITER_SEND = (char) 0x1;
    private static final char DELIMITER_RECEIVE = (char) 0x1;


    /**
     * Create the midlet.
     */
    public MobileControlMidlet() {
        super();
```

```
        gaugeForm.append(progressbar);

        alert = new Alert("", "", null, AlertType.INFO);
        alert.setTimeout(2000);

        try {
            addrBook = RecordStore.openRecordStore("TheAddressBook", true);
        } catch (RecordStoreException e) {
            addrBook = null;
        }

    }

    /*
     * (non-Javadoc)
     *
     * @see javax.microedition.midlet.MIDlet#startApp()
     */
    protected synchronized void startApp() throws MIDletStateChangeException {

        setStatus(STATUS_INITIAL);
        updateMidlet();

        readSettings();

        if (addrBook == null) {
            displayAlert(ERROR, "Could not open address book", null);
        } else {
            genMainScr();
        }


    }

    /**
     * Update the GUI of the midlet
     */
    private synchronized void updateMidlet() {
        listDisplay.setTicker(new Ticker("updateMidlet"));
        switch (getStatus()) {
            case STATUS_INITIAL:

                if (connection != null) {
                    closeConnection();
                } else {
                    displayMenu();
                }
                break;

            case STATUS_READ_SETTINGS:

                displayProgressBar("Read Settings", cmdBreak);

                break;

            case STATUS_SEARCH_URL:

                displaySearchUrl();
                break;

            case STATUS_WRITE_SETTINGS:
```

```
            displayProgressBar("Write Settings", cmdBreak);
            break;

        case STATUS_OPEN_SERIAL_CONNECTION:

            displayProgressBar("Open connection", cmdExit);
            break;

        case STATUS_CLOSE_APPLICATION:
        case STATUS_CLOSE_SERIAL_CONNECTION:

            displayProgressBar("Close connection", cmdExit);
            break;

        case STATUS_REQUEST_APPS:

            displayProgressBar("Request Applications", cmdBreak);
            break;

        case STATUS_SELECT_APP:

            displaySelectAppList(null);
            break;

        case STATUS_REQUEST_CONFIG:

            displayProgressBar("Request Actions", cmdBreak);
            break;

        case STATUS_CONTROLLING:
            listDisplay.setTicker(new Ticker("STATUS_CONTROLLING"));
            displaySelectActionList(null);
            break;

        case STATUS_SEARCH:
            //showInfo("No names found on the phone! \n\n"+ cevap);

            displaySearch();
            break;
        case STATUS_DISPLAY_SEARCH:
            showInfo("Nothing found on the phone! \n\n"+
                    "Google Desktop found "+cevap
                    + " results on the computer screen.\n\nThank you.");
            setStatus(STATUS_CONTROLLING);
            break;

        case STATUS_BROWSE:

            displayNames("Browse",null,null);
            break;

        case STATUS_ADD_NEW:

            displayAddNew();
            break;

       case STATUS_SEND_SHUTDOWN:

            displayProgressBar("Send Shutdown", cmdExit);
            break;

    }
```

```
    }

    /**
     * Display a list with all actions to control the current application.
     */
    private void displaySearch() {

        setStatus(STATUS_SEARCH);

        if (searchScr == null) {
           searchScr = new Form("Search");
            //searchScr.addCommand(cmdSearchNetwork);
            searchScr.addCommand(cmdBack);
            //searchScr.addCommand(cmdSearchLocal);
            searchScr.addCommand(cmdSearch);
            searchScr.setCommandListener(this);
            s_firstName = new TextField("First name:", "", FN_LEN,
                    TextField.ANY);
            s_lastName = new TextField("Last name:", "", LN_LEN,
TextField.ANY);
            searchScr.append(s_firstName);
            searchScr.append(s_lastName);
        }

        s_firstName.delete(0, s_firstName.size());
        s_lastName.delete(0, s_lastName.size());

        final Display display = Display.getDisplay(this);
       display.setCurrent(searchScr);
   }

    /**
     * Display a list with all actions to control the current application.
     */
    private void displaySelectActionList(final Vector actions) {

        setStatus(STATUS_CONTROLLING);

        if (actions != null) {
            // first time in this view => display actions & request title
            listDisplay.setNewDisplayEntries(actions);
            requestAppTitle();
        }

        // add the back command
        listDisplay.addCommand(cmdBack);
       // remove about cmd
        listDisplay.removeCommand(cmdAbout);

        // set title
        listDisplay.setTitle("Address Book - Menu");

        final Display display = Display.getDisplay(this);
        display.setCurrent(listDisplay);

   }

    /**
     * Display a list with all available applications.
     */
    private void displaySelectAppList(final Vector apps) {
```

```
        setStatus(STATUS_SELECT_APP);

        if (apps != null) {
            listDisplay.setNewDisplayEntries(apps);
            listDisplay.setTicker(null);
        }

        // set title
        listDisplay.setTitle("Apps");
        // add the back command
        listDisplay.addCommand(cmdBack);
        // remove about cmd
        listDisplay.removeCommand(cmdAbout);
        listDisplay.removeCommand(cmdExit);

        final Display display = Display.getDisplay(this);
        display.setCurrent(listDisplay);

    }

    /**
     * Display a list with all available applications.
     */
    private void displayAddressMenu(final Vector apps) {

        setStatus(STATUS_CONTROLLING);

        if (apps != null) {
            listDisplay.setNewDisplayEntries(apps);
            listDisplay.setTicker(null);
        }

        // set title
        listDisplay.setTitle("Apps");
        // add the back command
        listDisplay.addCommand(cmdBack);
        // remove about cmd
        listDisplay.removeCommand(cmdAbout);

        final Display display = Display.getDisplay(this);
        display.setCurrent(listDisplay);

    }

    /**
     * Display a progress bar.
     *
     * @param title
     */
    private void displayProgressBar(final String title, final Command cmd) {

        progressbar.setLabel(title);

        if (cmd != null) {
            gaugeForm.addCommand(cmd);
            gaugeForm.setCommandListener(this);
        }

        final Display display = Display.getDisplay(this);
        display.setCurrent(gaugeForm);
    }
```

```
/**
 * Display the general menu with "Search Device" and "Connect to ...".
 *
 * @param commands
 * @param displayEntires
 */
private void displayMenu() {

    final Vector displayEntries = new Vector();
    displayEntries.addElement("Search Device");

    if (connection != null) {
        displayEntries.addElement("Close " + deviceName);
   } else if (url != null && url.trim().length() > 0) {
        displayEntries.addElement("Connect to " + deviceName);
    }

    listDisplay.setNewDisplayEntries(displayEntries);
    // this view contains max 2 (start search/select url) entries and
    // default selection is set to second entry, as that will contain the
    // 'url' item.
    if (displayEntries.size() > 1) {
        listDisplay.setSelectedIndex(1, true);
    }

    // be sure to remove ticker
    listDisplay.setTicker(null);

    // remove the back command
    listDisplay.removeCommand(cmdBack);

    // add about cmd
    listDisplay.removeCommand(cmdAbout);
    listDisplay.addCommand(cmdExit);
    listDisplay.addCommand(cmdSelect);

    //set title
    listDisplay.setTitle("Address Book");

    final Display display = Display.getDisplay(this);
    display.setCurrent(listDisplay);
}

/**
 * Start the serial connection.
*/
private synchronized void startConnection() {

    if (connection != null) {
        // another connection still open
        // closeConnection();
        showError("another connection still open");
    } else {
        openConnection();
    }

}

/**
 * Read the stored connection settings rom RMS.
 *
 * @return
```

```
 */
private synchronized void readSettings() {

    setStatus(STATUS_READ_SETTINGS);
    updateMidlet();

    final SettingStorage storage = new SettingStorage();
    try {
        storage.read();
        url = storage.getUrl();
        deviceName = storage.getDeviceName();

    } catch (RecordStoreException e) {

        // ignore error
    }

    setStatus(STATUS_INITIAL);
    updateMidlet();

}

/**
 * Write the retrieved connection settings rom RMS.
 *
 * @return
 */
private synchronized void writeSettings() {

    setStatus(STATUS_WRITE_SETTINGS);
    updateMidlet();

    final SettingStorage storage = new SettingStorage();
    try {
        storage.setUrl(url);
        storage.setDeviceName(deviceName);
        storage.write();

    } catch (RecordStoreException e) {
        // ignore errors
    }

}

/*
 * @see javax.microedition.midlet.MIDlet#pauseApp()
 */
protected void pauseApp() {
    // do nothing

}

/*
 * @see javax.microedition.midlet.MIDlet#destroyApp(boolean)
 */
protected synchronized void destroyApp(boolean arg0)
throws MIDletStateChangeException {

    sendShutDown();

}
```

```
    /*
     * (non-Javadoc)
     */
    public synchronized void urlDiscovered(final String url, final String name)
{

        this.url = url;
        this.deviceName = name;

        writeSettings();

        startConnection();

    }

    /*
     * (non-Javadoc)
     */
    public synchronized void errorOccuredDuringConnectionSearch(
            final String message) {

        setStatus(STATUS_INITIAL);

        connected = false;
        updateMidlet();
        showError(message);
    }

    private synchronized void showError(final String msg) {

        final Alert alert = new Alert("Error", msg, null, AlertType.ERROR);
        alert.setCommandListener(this);
        alert.setTimeout(Alert.FOREVER);
       Display.getDisplay(this).setCurrent(alert);

    }

    private synchronized void showInfo(final String msg) {

        final Alert alert = new Alert("Info", msg, null, AlertType.INFO);
        alert.setCommandListener(this);
        alert.setTimeout(Alert.FOREVER);
        Display.getDisplay(this).setCurrent(alert);

    }

    /**
     * Select a new connection.
     */
    private synchronized void selectNewDevice() {

        setStatus(STATUS_SEARCH_URL);
        updateMidlet();

        try {
            connectScreen.startUrlSelection(this, null, true);
        } catch (IOException e) {
            setStatus(STATUS_INITIAL);
            updateMidlet();
            showError("Device selection failed (" + e.toString() + ")");
        }
    }
```

```
/**
 * Display the BT search device screen(s).
 */
private void displaySearchUrl() {

    final Display display = Display.getDisplay(this);
    display.setCurrent(connectScreen);
}

/*
 * (non-Javadoc)
 */
public synchronized void connectionClosed() {

    connected = false;
    connection = null;

    if (getStatus() == STATUS_CLOSE_APPLICATION) {
        notifyDestroyed();
    } else {
        setStatus(STATUS_INITIAL);
        updateMidlet();
    }

}

/**
 * Open the serial connection.
 */
private synchronized void openConnection() {

    setStatus(STATUS_OPEN_SERIAL_CONNECTION);
    updateMidlet();

    connection = new SerialConnection(this, nullLogger, url,
            ServiceRecord.NOAUTHENTICATE_NOENCRYPT);
}

/*
 * (non-Javadoc)
 */
public synchronized void connectionOpened() {

    connected = true;

    requestApplications();

}

/**
* Request all available applications. Format: "QAP"
 */
private synchronized void requestApplications() {

    setStatus(STATUS_REQUEST_APPS);
    updateMidlet();
    sendMessage("QAP");

}

/**
```

```
     * Request current application title. Format: "QAT0x2appname"
     */
    private synchronized void requestAppTitle() {

        if (getStatus() == STATUS_CONTROLLING) {
            sendMessage("QAT" + DELIMITER_SEND + currentApplication);
        }
    }

    /*
     * (non-Javadoc)
     */
    public synchronized void errorOccuredDuringConnection(final String message)
{

        connected = false;
        connection = null;
       if (getStatus() == STATUS_CLOSE_APPLICATION) {
            notifyDestroyed();
        } else {
            setStatus(STATUS_INITIAL);
            updateMidlet();
            showError(message);
        }
    }

    /*
     * (non-Javadoc)
     */
    public synchronized void messageSendSucceeded(int msgId) {

        if (getStatus() == STATUS_CLOSE_APPLICATION
                || getStatus() == STATUS_SEND_SHUTDOWN) {
            // last message was "QAS"
            closeConnection();
        }

        // ignore other messages confirmations
    }

    /*
     * (non-Javadoc)
     */
    public synchronized void receiveMessage(final String message) {

        if (message.length() > 3) {
            cevap = message.substring(3);
        }

        if (getStatus() == STATUS_CONTROLLING
                && (message.startsWith("RAT") || message.startsWith("RAA"))) {

            receiveNewTitle(message);

        } else if (getStatus() == STATUS_REQUEST_APPS
                && message.startsWith("RAP")) {

            receiveApplications(message);

        } else if (getStatus() == STATUS_REQUEST_CONFIG
                && message.startsWith("RAC")) {
```

```
            receiveConfiguration(message);

        }

    }

    /**
     * Receive a new title. Message format "RATtitle".
     *
     * @param message
     */
    private void receiveNewTitle(final String message) {

        if (message.length() > 3) {
            listDisplay.setTicker(new Ticker(message.substring(3)));
            listDisplay.setTicker(null);
        }
    }

    /**
     * Receive a new configuration with an action list for the current
     * application. Format: "RACaction0x1action0x1"
     *
     * @param message
     */
    private synchronized void receiveConfiguration(final String message) {

        if (message.length() > 3) {
            final Vector actions = getStrings(message.substring(3),
                    DELIMITER_RECEIVE);

            displaySelectActionList(actions);
        }
    }

    /**
     * Return String elements tokenized excluding delimiter.
     *
     * @param string
     *            the string to tokenize
     * @param delimiter
     *            the char to overread
     * @return Vector of String
     */
    private static Vector getStrings(String string, char delimiter) {

        Vector result = new Vector();
        int delimPos = 0;
        while (delimPos >= 0) {
            int nextDelimPos = string.indexOf(delimiter, delimPos);
            if (nextDelimPos >= 0) {
                String element = string.substring(delimPos, nextDelimPos);
                result.addElement(element);
            } else {
                return result;
           }
            delimPos = nextDelimPos + 1;
        }
        return result;
    }

    /**
```

```
 * Receive a new list of available applications. Format:
 * "RAPaction0x1action0x1"
 *
 * @param message
 */
private synchronized void receiveApplications(String message) {

    if (message.length() > 3) {
        final Vector apps = getStrings(message.substring(3),
                DELIMITER_RECEIVE);

        displaySelectAppList(apps);
   }
}

/**
 * Send a test message
 *
 * @param string
 */
private synchronized void sendMessage(final String msg) {

    if (connection != null && connected) {
        connection.sendMessage(msg);
    }

    updateMidlet();

}

/**
 * Close the serial connection.
 */
private void closeConnection() {

    if (connection == null && getStatus() == STATUS_CLOSE_APPLICATION) {
        notifyDestroyed();
        return;
    }
    if (connection != null) {
        if (getStatus() != STATUS_CLOSE_APPLICATION) {
            setStatus(STATUS_CLOSE_SERIAL_CONNECTION);
       }
        connected = false;
        updateMidlet();
        connection.close();

    }

}

public synchronized void searchAborted() {

    setStatus(STATUS_INITIAL);

    connected = false;
    updateMidlet();

}

/**
 * A command was executed.
```

```
     *
     * @param command
     * @param screen
     * @see ...CommandListener#commandAction(javax.microedition.lcdui.Command,
     *      javax.microedition.lcdui.Displayable)
     */
    public synchronized void commandAction(final Command command,
            final Displayable screen) {

        if (getStatus() == STATUS_CLOSE_APPLICATION) {
            // DO NOT ALLOW ANY NEW COMMANDS
            return;
        }

        if (command == Alert.DISMISS_COMMAND) {
            // Alert closed

            updateMidlet();

        } else if (command == cmdBack || command == cmdBreak || command ==
cmdCancel) {
            // BACK

            if (getStatus() == STATUS_REQUEST_CONFIG
                    || getStatus() == STATUS_CONTROLLING) {
                requestApplications();
            } else if (getStatus() == STATUS_SELECT_APP) {
                sendShutDown();
            } else if (getStatus() == STATUS_ADD_NEW) {
                setStatus(STATUS_CONTROLLING);
                updateMidlet();
            } else if (getStatus() == STATUS_BROWSE) {
                setStatus(STATUS_CONTROLLING);
                updateMidlet();
            } else if (getStatus() == STATUS_SEARCH) {
                setStatus(STATUS_CONTROLLING);
                updateMidlet();
            } else {
                setStatus(STATUS_INITIAL);
                updateMidlet();
            }

        } else if (command == cmdExit) {
            // EXIT

            closeApp();

        } else if (command == cmdAbout) {
            // show about info
            showAbout();

        } else if ((getStatus() == STATUS_BROWSE) || (command == cmdDial)){

            showAbout();

        } else if ((getStatus() == STATUS_ADD_NEW) || (command == cmdAdd)){

            displayAddConfirm();

        } else if (command == cmdSearch) {
            // SELECT APP
            listDisplay.setTicker(new Ticker("command == cmdSearch"));
```

```
            // display search of local addr book
            displayNames("Search Result", s_firstName.getString(),
s_lastName.getString());

        } else if (getStatus() == STATUS_SELECT_APP) {
            // SELECT APP

            currentApplication = listDisplay.getString(listDisplay
                    .getSelectedIndex());
            applicationSelected();

        }  else if (getStatus() == STATUS_CONTROLLING) {
            // SELECT ACTION

            final String action = listDisplay.getString(listDisplay
                    .getSelectedIndex());
            actionSelected(action, command, screen);

        } else if (getStatus() == STATUS_INITIAL) {
            // MENU SCREEN

            final int index = listDisplay.getSelectedIndex();
            switch (index) {
                case 0: // "Search Device"
                    selectNewDevice();
                    break;
                case 1: // "Connect to ..."
                    startConnection();
                    break;
            }

        }
    }

    /**
     * Show About information
     */
    private void showAbout() {
        showInfo("MobilePC\nMobilePC created in 2006 by Mehmet AKBAS");

    }

    /**
     * Send a SHUTDOWN message
     */
    private void sendShutDown() {

        if (getStatus() != STATUS_CLOSE_APPLICATION) {
            setStatus(STATUS_SEND_SHUTDOWN);
        }
        updateMidlet();
        sendMessage("QAS");
    }

    /**
     * Search screen.
     *
     * Displays two <code>TextField</code>s: one for first name, and one for
     * last name. These are used for searching the address book.
     *
     * @see AddressBookMIDlet#genNameScr
```

```
     */
    private Screen genSearchScr() {
        if (searchScr == null) {
            searchScr = new Form("Search");
            searchScr.addCommand(cmdSearchLocal);
            searchScr.addCommand(cmdSearchNetwork);
            searchScr.addCommand(cmdBack);
            searchScr.setCommandListener(this);
            s_firstName = new TextField("First name:", "", FN_LEN,
                    TextField.ANY);
            s_lastName = new TextField("Last name:", "", LN_LEN,
TextField.ANY);
            searchScr.append(s_firstName);
            searchScr.append(s_lastName);
        }

        s_firstName.delete(0, s_firstName.size());
        s_lastName.delete(0, s_lastName.size());

        display.setCurrent(searchScr);
        return searchScr;
    }


    /**
     * Generates a list of first/last/phone numbers. Can be called as a result
     * of a browse command (genBrowseScr) or a search command (genSearchScr).
     *
     * title title of this screen (since it can be called from a browse or a
     * search command. f if not null, first name to search on l if not null,
     * last name to search on
     */
    private void displayNames(String title, String f, String l) {
        SimpleComparator sc;
        SimpleFilter sf = null;
        RecordEnumeration re = null;
        phoneNums = null;

        sc = new SimpleComparator(
                sortOrder == 0 ? SimpleComparator.SORT_BY_FIRST_NAME
                : SimpleComparator.SORT_BY_LAST_NAME);

        if (f != null || l != null) {
            sf = new SimpleFilter(f, l);
        }

        try {
            re = addrBook.enumerateRecords(sf, sc, false);
        } catch (Exception e) {
            showError("Could not create enumeration: " + e);
            //  return null;
        }

        nameScr = null;
        if (re.hasNextElement()) {
            nameScr = new List(title, List.IMPLICIT);
            nameScr.addCommand(cmdBack);
            nameScr.addCommand(cmdDial);
            nameScr.setCommandListener(this);
            phoneNums = new Vector(6);

            try {
```

```
                while (re.hasNextElement()) {
                    byte[] b = re.nextRecord();
                    String pn = SimpleRecord.getPhoneNum(b);
                    nameScr.append(SimpleRecord.getFirstName(b) + " "
                            + SimpleRecord.getLastName(b) + " "
                            + SimpleRecord.getPhoneNum(b), null);
                    phoneNums.addElement(pn);
                }
            } catch (Exception e) {
                showError("Error while building name list: " + e);
                //  return null;
            }

            final Display display = Display.getDisplay(this);
            display.setCurrent(nameScr);

        } else {
            final String msg = "QAA" + f + " " + l + DELIMITER_SEND +
currentApplication;
            sendMessage(msg);

            setStatus(STATUS_DISPLAY_SEARCH);
            updateMidlet();
            connection.handleReceivedMessage()onListener .c
.handleReceivedMessage(new byte[] msageBytes);
            showInfo("No names found on the phone! \n\n"+ cevap);
        }


    }

    /**
     * Add an entry to the address book. Called after the user selects the
     * addCmd while in the genEntryScr screen.
     */
    private void displayAddNew() {
        if (entryScr == null) {
            entryScr = new Form("Add new");
            entryScr.addCommand(cmdCancel);
            entryScr.addCommand(cmdAdd);
            entryScr.setCommandListener(this);

            e_firstName = new TextField("First name:", "", FN_LEN,
                    TextField.ANY);
            e_lastName = new TextField("Last name:", "", LN_LEN,
TextField.ANY);
            e_phoneNum = new TextField("Phone Number", "", PN_LEN,
                    TextField.PHONENUMBER);
            entryScr.append(e_firstName);
            entryScr.append(e_lastName);
            entryScr.append(e_phoneNum);
        }

        e_firstName.delete(0, e_firstName.size());
        e_lastName.delete(0, e_lastName.size());
        e_phoneNum.delete(0, e_phoneNum.size());

        final Display display = Display.getDisplay(this);
        display.setCurrent(entryScr);
    }

    private void displayAddConfirm() {
```

```
        String f = e_firstName.getString();
        String l = e_lastName.getString();
        String p = e_phoneNum.getString();

        byte[] b = SimpleRecord.createRecord(f, l, p);

        try {
            addrBook.addRecord(b, 0, b.length);
            showInfo("Record added");
            setStatus(STATUS_CONTROLLING);
            //updateMidlet();

        } catch (RecordStoreException rse) {
            showError("Could not add record" + rse);
        }
    }


    /**
     * A new action is selected => send to serial connection. Format:
     * "QAAaction0x2appname"
     *
     * @param action
     */
    private synchronized void actionSelected(final String action, Command c,
Displayable d) {

        if ((c == List.SELECT_COMMAND) || (c == cmdSelect)) {
            if (action.equals("Search")) {
                //showInfo("Search");
                //displaySearch();
                setStatus(STATUS_SEARCH);
                  final String msg = "QAA" + action + DELIMITER_SEND +
currentApplication;
                  sendMessage(msg);
            } else if (action.equals("Browse")) {
                //showInfo("Browse");
                setStatus(STATUS_BROWSE);
            } else if (action.equals("Add New")) {
                //showInfo("Add_New");
                setStatus(STATUS_ADD_NEW);
            } else
                setStatus(STATUS_CONTROLLING);
        }

        updateMidlet();

    }

   /**
     * A new action is selected => send to serial connection to get
     * configuration. Format: "QAC0x2appname"
     *
     */
    private synchronized void applicationSelected() {

        setStatus(STATUS_REQUEST_CONFIG);
        final String msg = "QAC" + DELIMITER_SEND + currentApplication;
        sendMessage(msg);

    }
```

```
/**
 * Close the application.
 */
private synchronized void closeApp() {

    setStatus(STATUS_CLOSE_APPLICATION);
    if (connection != null) {
        sendShutDown();
    } else {
        notifyDestroyed();
    }

}

/**
 * Get the status of the midlet.
 *
 * @return
 */
private int getStatus() {
    return status;
}

/**
 * Set the status of the midlet.
 *
 * @param newStatus
*/
private synchronized void setStatus(final int newStatus) {

    if (getStatus() != STATUS_CLOSE_APPLICATION) {
        //do not change status during closing of application

        this.status = newStatus;
    }
}

/**
 * No log screen for this application.
 *
 */
public void viewLogScreen() {
    // ignore

}

/**
 * Display an Alert on the screen
 *
 * @param type : One of the following: ERROR, INFO
 * @param msg :  Message to display
 * @param s : screen to change. if null, revert to main screen
 */
private void displayAlert(int type, String msg, Screen s) {
    alert.setString(msg);

    switch (type) {
        case ERROR:
            alert.setTitle("Error!");
            alert.setType(AlertType.ERROR);
            break;
```

```
            case INFO:
                alert.setTitle("Info");
                alert.setType(AlertType.INFO);
                break;
        }
        final Display display = Display.getDisplay(this);
        display.setCurrent(alert, s == null ? display.getCurrent() : s);
    }


}
```

## B. PC

### 1. OVERVIEW

The software that runs on personal computer simply serves as a listener when a search is initiated at the mobile phone. If a query is received from the mobile phone the query is translated into a proper form for Google Desktop application, and the search results are sent back to the mobile phone. If the query contains related flags which trigger Internet search, Google desktop application extends the search to the Web. If the search is started at the personal computer, this application first searches the local device and then extends the search to include mobile phone, if needed.

### 2. LIBRARIES AND APPLICATIONS

This application was created in Java. The integration between this application and Google Desktop is done with the help of GDS Java API [46]. It is a simple interface for access to Google desktop search from any java application.

### 3. PROGRAM CODE

The java code listed here is the class of the application, which processes the messages coming from the mobile phone.

```
/*
 ***************************************************************************
 *
 * Java Tools for common purposes.
 *
 ***************************************************************************/
package mobilepc.appcontrol;

import java.io.FileNotFoundException;
import java.io.FileReader;

import java.util.List;

import javax.swing.JOptionPane;
import javax.swing.UIManager;

import com.thoughtworks.xstream.XStream;
import com.thoughtworks.xstream.io.xml.DomDriver;

import mobilepc.appcontrol.log.Logger;
import mobilepc.appcontrol.remote.RemoteHandler;
import mobilepc.appcontrol.remote.RemoteHandlerException;

import javax.swing.*;
import java.awt.*;
import java.lang.*;
import java.awt.event.*;
import javax.swing.JCheckBoxMenuItem;
import java.util.Iterator;

import javax.xml.bind.JAXBException;

import jgd.JGDError;
import jgd.JGDQuery;
import jgd.Util;
import jgd.jaxb.Results;
import jgd.jaxb.ResultsType;

/**
 *
 * The RemoteControl type is the main class which allows com port remote
 * controlling.
 */
public class RemoteControl  extends JFrame implements MessageReciever {
    /**
     * Default frame width
     */
    private static final int FRAME_WIDTH = 640;


    /**
     * Default frame height
     */
    private static final int FRAME_HEIGHT = 480;


    /**
     * X coordinate of the frame default origin point
     */
    private static final int FRAME_X_ORIGIN = 100;
```

```
/**
 * Y coordinate of the frame default origin point
 */
private static final int FRAME_Y_ORIGIN = 60;

JTextArea jtOutput;

/**
 * Default config file name
 *
 * @see AppControlConfig for details.
 */
private static String CONFIG_FILE_NAME = "config.xml";

/** the version number */
public static String Version = "V1.2.1";

/**
 * Start the program.
 *
 * @param args
 *            first arg is used for xml config file name
 */
public static void main(String[] args) {
    // set landf
    try {
        UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    } catch (Exception e) {
    }
    // select file name for config file
    String configFileName = CONFIG_FILE_NAME;
    if (args.length >= 1) {
        configFileName = args[0];
    }
    try {
       RemoteControl frame = new RemoteControl(configFileName);
        frame.setVisible(true);

        // run control
        //       new RemoteControl(configFileName);
    } catch (Exception e) {
        System.err.println("Remote Control could not be run: " + e);
        showErrorDialog("Remote Control could not be run started\n" + e);
        System.exit(-1);

    } catch (Error e) {
        System.err.println("Remote Control could not be run: " + e);
        showErrorDialog("Remote Control could not be run started\n" + e);
        e.printStackTrace();
        System.exit(-1);
    }
}

/**
 * @param error
 */
public static void showAboutDialog() {
    JOptionPane
            .showMessageDialog(
            null,
            "Mobile Bluetooth Connection"
            + Version
```

```
            + "\n\nMobile device to use Bluetooth connection.\n",
           "About", JOptionPane.INFORMATION_MESSAGE);

}

/**
 * Show an error to the user.
 *
 */
private static void showErrorDialog(String error) {
    JOptionPane.showMessageDialog(null, error, "Problem occured",
            JOptionPane.ERROR_MESSAGE);

}

/** the connectivity to the remote device */
private RemoteHandler comms;

/** the configuration */
private AppControlConfig configuration;

/** the control instance to use for talking to the window system */
private AppControl control;

/** shutdown indicator */
private boolean shutdown;

/** simple gui */
private TrayInterface tray;

/** Log */
private Logger log = Logger.getInstance();

/**
 * Construct the type.
 *
 * @param configFile
 *            the config file name to use
 * @throws AppControlException
 *             is thrown if object could not be built.
 */
public RemoteControl(String configFile) throws AppControlException {
    super();
    jtOutput = new JTextArea(5, 20);
    jtOutput.setLineWrap(true);
    jtOutput.setWrapStyleWord(true);
    jtOutput.setEditable(false);
    jtOutput.setFont(new Font("Verdana", Font.BOLD, 12));
    JScrollPane scrollPane = new JScrollPane(jtOutput,
            JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
            JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
    getContentPane().add(scrollPane, java.awt.BorderLayout.CENTER);
    setTitle("Google Desktop - Mobile Search Results");
    setSize(FRAME_WIDTH, FRAME_HEIGHT);
    setResizable(true);
    setLocation(FRAME_X_ORIGIN, FRAME_Y_ORIGIN);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    jtOutput.append("->Waiting for the query?"+"\n");


    synchronized (this) {
        log.addLog(new String[] { "Mobile Control started. Version: ",
```

```
        Version });
        initializeConfiguration(configFile);
        initializeControl();
        initializeCommunication();
    }
    initializeTray();
    registerShutdownHook();
}

/**
 * Search matching Application Interaction for the message recieved.
 *
 * @return an AppInteraction to use or first if none found to match
 */
private AppInteraction getInteraction(RemoteMessage msg) {
    // search for name matches
    return configuration.getAppInteractionByName(msg
            .getApplicationIdentifier());
}

/**
 * Do the action.
 *
 * @param msg
 */
private void handleRequestAction(RemoteMessage msg) {
    // locate app
    AppInteraction interaction = getInteraction(msg);


    String result = "";
    //log
    log.addLog(new String[] { "RecievedMessage: Action ", msg.getPayLoad(),
    " on ", interaction.getNameForDisplay() });

    // call app
    try {
        if (interaction != null) {
            result = interaction.invokeAction(control, msg.getPayLoad());
            //log
            log.addLog(new String[] { "SendMessage: Action for ",
            interaction.getNameForDisplay(), " is ", result });
        } else {
            result = msg.getApplicationIdentifier()
            + " not found in config.";
            tray.showMessage("Problem occured", result);
            //log
            log.addLog(new String[] { "RecievedMessage: Action failed ",
            result });

        }
    } catch (AppControlException e) {
        result = interaction.getNameForDisplay()
        + " > Action failed. Check applications and retry";
        tray.showMessage("Problem occured", result);
        //log
        log
                .addLog(new String[] { "RecievedMessage: Action failed ",
                result });
        e.printStackTrace();
        rebuildAccess();
    }
```

```
        try {
            // send back new title.
            comms.sendMessage(RemoteMessage.createReplyAction(result));
        } catch (RemoteHandlerException e) {
            e.printStackTrace();
            tray.showMessage("Problem occured: ", e.toString());
            //log
            log.addLog(new String[] { "RecievedMessage: Action failed ",
            e.toString() });
        }
    }

    /**
     * Deliver remote control actions for an application
     *
     * @param msg
     */
    private void handleRequestAppConfig(RemoteMessage msg) {
        AppInteraction interaction = getInteraction(msg);
       if (interaction == null) {
            String result = msg.getApplicationIdentifier()
            + " not found in config.";
            tray.showMessage("Problem occured", result);
            //log
            log.addLog(new String[] { "RecievedMessage: GetConfig ", result });
            return;
        }
        //log
        log.addLog(new String[] { "RecievedMessage: GetConfig "
                + interaction.getNameForDisplay() });

        // singal tray
        tray.setConnected(true, interaction.getNameForDisplay());
        tray.showMessage("Control acquired", interaction.getNameForDisplay());

        try {
            // send Actions
            comms.sendMessage(RemoteMessage.createReplyConfig(interaction
                    .getAvailableActions()));
            //log
            log.addLog(new String[] { "SendMessage: GetConfig ",
            interaction.getNameForDisplay(), " ",
            interaction.getAvailableActions().toString() });
        } catch (RemoteHandlerException e) {
            e.printStackTrace();
            tray.showMessage("Problem occured: ", e.toString());
            //log
            log.addLog(new String[] { "RecievedMessage: GetConfig failed ",
            e.toString() });
        }
    }

    /**
    * Retrieve all configured application sets.
     */
    private void handleRequestApplications() {
        //log
        log.addLog("RecievedMessage: GetApps ");
        String[] logStrings = new String[configuration.applications.length + 1];
        logStrings[0] = "SendMessage: GetApps: ";
        for (int i = 0; i < configuration.applications.length; i++) {
            logStrings[i + 1] = configuration.applications[i]
```

```
                .getNameForDisplay();
        }
        log.addLog(logStrings);

        // send data
        try {
            comms.sendMessage(RemoteMessage
                    .createReplyApplication(configuration.applications));
        } catch (RemoteHandlerException e) {
            e.printStackTrace();
            tray.showMessage("Problem occured: ", e.toString());
           //log
            log.addLog(new String[] { "RecievedMessage: GetApps failed ",
            e.toString() });
        }
    }

    /**
     * Retrieve application title.
     */
    private void handleRequestAppTitle(RemoteMessage msg) {

        String title = "";
        // get Interaction from message.
        AppInteraction interaction = getInteraction(msg);
        //log
        log.addLog(new String[] { "RecievedMessage: GetTitle for ",
        interaction.getNameForDisplay() });
        try {
            // retrieve title
            if (interaction != null) {
                title = control.getCurrentTitle(interaction.getTitleRegex());
                //log
                log.addLog(new String[] { "SendMessage: GetTitle for ",
                interaction.getNameForDisplay(), " is ", title });
            } else {
                title = msg.getApplicationIdentifier()
                + " not found in config.";
                tray.showMessage("Problem occured", title);
                //log
                log.addLog(new String[] {
                    "RecievedMessage: GetTitle failed:  ", title });
            }

        } catch (AppControlException e1) {
            title = interaction.getNameForDisplay()
            + " > Title not available. Check applications.";
            tray.showMessage("Problem occured", title);
            //log
            log.addLog(new String[] { "RecievedMessage: GetTitle failed: ",
            title, e1.toString() });
            e1.printStackTrace();
            rebuildAccess();
        }
        try {
            comms.sendMessage(RemoteMessage.createReplyCurrentTitle(title));
        } catch (RemoteHandlerException e) {
            e.printStackTrace();
            tray.showMessage("Problem occured: ", e.toString());
            //log
            log.addLog(new String[] { "RecievedMessage: GetTitle failed ",
            e.toString() });
```

```
    }
}

/**
 * Init communication to the remote system.
 */
private void initializeCommunication() throws AppControlException {
    try {
        comms = new RemoteHandler(this, configuration.portNumber);
    } catch (RemoteHandlerException e) {
        throw new AppControlException(
                "Remote Handler could not be initialized. Check port.", e);
    }
}

/**
 * Read config file to build configuration.
 *
 * @param configFile
 *            filename to read from
 * @throws AppControlException
 */
private void initializeConfiguration(String configFile)
throws AppControlException {

    XStream xstream = new XStream(new DomDriver());
    xstream.alias("appInteraction", GenericInteraction.class);
    xstream.alias("keyAction", KeyAction.class);
    xstream.alias("AppControlConfig", AppControlConfig.class);

    try {
        FileReader fr = new FileReader(configFile);
        configuration = (AppControlConfig) xstream.fromXML(fr);
        configuration.init();
        log.addLog("Config init. Port COM" + configuration.portNumber);
    } catch (FileNotFoundException e) {
        throw new AppControlException("File Not Found: " + configFile);
    } catch (AppControlException e) {
        throw e;
    } catch (Exception e) {
        throw new AppControlException(
                "Config could not be read. Check XML File structure. ("
                + e.toString() + ")", e);
    }

}

/**
 * Set the configured Control Class.
 *
 * @throws AppControlException
 *             if Control Class could not be build
 *
 */
private void initializeControl() throws AppControlException {
    Class clazz;
    try {
        clazz = Class.forName(configuration.appControlClass);
        control = (AppControl) clazz.newInstance();
        // lets see, what is running
        List titles = control.getActiveTitles();
        log.addLog(new String[] { "Control init. Titles: ",
```

```
            titles.toString() });
        } catch (Exception e) {
            throw new AppControlException("Configured control class "
                    + configuration.appControlClass + " could not be created",
                    e);
        } catch (Error e) {
            throw new AppControlException("Configured control class "
                    + configuration.appControlClass + " could not be created",
                    e);
        }

    }

    /**
     * Init the Tray.
     */
    private void initializeTray() {
        tray = new TrayInterface(this);
    }

    /**
     * Work an Error and shutdown application.
     *
     * @param error
     *            the String explaining the error
     * @param t
     *            the Throwable causing the Error
     */
    public void processError(String error, Throwable t) {
        showErrorDialog(error);
        shutdown();
    }

    /**
     * process a message as Message Reciever.
     *
     * @param msg -
     *            the remote message to check
     */
    public synchronized void processMessage(RemoteMessage msg)   {
        String queryString = new String(msg.getPayLoad());
        if (msg.getType() == RemoteMessage.REQUEST_ACTION) {
            System.out.println(queryString);
            jtOutput.append("\n" + "->Searching : " + queryString + "\n");
            Results r = null;
            try {
                if (queryString.length() <= 1)
                    queryString = "bos";
                JGDQuery q = new JGDQuery(queryString);
                q.setNum( new Integer( 25 ) );
                q.setStart( new Integer( 0 ) );
                q.setSortedByRelevance() ;
                q.setFilterByFiles() ;
                q.setFileType("doc");

                r = q.execute();

                // send data
                try {
                    comms.sendMessage(msg.createReplyAction(" " +
r.getCount()));
                } catch (RemoteHandlerException e) {
```

```
                    e.printStackTrace();
                    tray.showMessage("Problem occured: ", e.toString());
                }


                //System.out.println();

                //System.out.println( "First 25 of Results:"  + r.getCount());

               if (r.getCount() == 0){
                    jtOutput.append("\n" + "->Nothing found!" + "\n");

                } else if (r.getCount() < 25){
                    jtOutput.append("\n" + "->First " + r.getCount()+ " of a
total of "  + r.getCount() + " results:\n");
                } else {
                    jtOutput.append("\n" + "->First 25 of a total of "  +
r.getCount() + " results:\n");
                }

                //System.out.println();
                jtOutput.append( "---------------------" + "\n");

                List l = r.getResult();
                for (Iterator i = l.iterator(); i.hasNext();) {
                    ResultsType.ResultType element = (ResultsType.ResultType)
i.next();
                    jtOutput.append( element.getUrl() + "\n");

                }
                jtOutput.append( ==================================" + "\n");
                jtOutput.setCaretPosition(jtOutput.getText().length());
            } catch (JGDError ex) {
                ex.printStackTrace();
            }
        }


        // evaluateType
        if (msg.getType() == RemoteMessage.REQUEST_APPS) {
            handleRequestApplications();
        } else if (msg.getType() == RemoteMessage.REQUEST_SHUTDOWN) {
            // means remote device will not sent anything anymore, so disconnect
            tray.setConnected(false, null);
            //log
            log.addLog("RecievedMessage: Shutdown");
        } else if (msg.getType() == RemoteMessage.REQUEST_APP_CURRENT_TITLE) {
            handleRequestAppTitle(msg);
        } else if (msg.getType() == RemoteMessage.REQUEST_APP_CONFIG) {
            handleRequestAppConfig(msg);
        } else if (msg.getType() == RemoteMessage.REQUEST_ACTION) {
            // handleRequestAction(msg);
        } else {
            log.addLog(new String[] { "RecievedMessage: Unknown Message ",
            msg.getPayLoad() });
        }
    }

    /**
     * Renew control of window system.
     */
    private void rebuildAccess() {
```

```
            try {
                 List titles = control.renewActiveTitles();
                 log.addLog(new String[] { "Control Renew. Titles: ",
                 titles.toString() });
             } catch (AppControlException e2) {
                 e2.printStackTrace();
                 log.addLog(new String[] { "Control Renew. Exception: ",
                 e2.toString() });
             }
        }

        /**
         * use shutdown hook for shutting down connectivity.
        */
        private void registerShutdownHook() {
            Thread hook = new Thread("RemoteControl Shutdown Hook") {
                public void run() {
                    RemoteControl.this.shutdown();
                }
            };
            Runtime.getRuntime().addShutdownHook(hook);
        }

        /**
         * Shutdown the program.
         */
        public void shutdown() {
            // shutdown once only
            if (shutdown)
                return;
            shutdown = true;
            // tray down
            if (tray != null) {
                tray.shutDown();
            }

            // comms down
            synchronized (this) {
                if (comms != null)
                    comms.shutdown();
            }
            // exit
            System.exit(0);
        }

    }
```

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES


[1] J. Teevan, W. Jones, and B. B. Bederson, *"Personal Information Management"*, Communications Of ACM, vol.49, no.1, pp. 40-43, January 2006.

[2] R. Boardman, *"Improving Tool Support for Personal Information Management",* Ph.D. Thesis, Department of Electrical and Electronic Engineering, Imperial College London and University of London, July 2004.

[3] M. Lansdale, *"The Psychology Of Personal Information Management"*, Applied Ergonomics 19, pp. 55–66, 1998.

[4] W.F. Jones, *"Keepers? The Present and Future Perfect In Support Of Personal Information Management",* First Monday 9, p. 3, March 2004.

[5] E. Cutrell, *"Searching to Eliminate Personal Information Management"*, Communications of ACM, vol.49, no.1, January 2006.

[6] G. Singh, *"Mobile Devices",* Course Slides for CS4135, Department of Computer Science, Naval Postgraduate School, April 2007.

[7] T. Hammond, T. Hannay, and B. Lund, *"The Role of RSS in Science Publishing"*, D-Lib Magazine, vol. 10, pp. 1082-9873, 2004.

[8] B.S. Manjunath, P. Salembier, and T. Sikora, *"Introduction to MPEG-7: Multimedia Content Description Interface"*, Wiley & Sons, April 2002.

[9] M. Czerwinski, D.W. Gage, J. Gemmell, C.C. Marshall, M.A. Pérez-Quiñonesis, M.M. Skeels, and T. Catarci, *"Digital Memories in an Era of Ubiquitous Computing and Abundant Storage"*, Communications Of ACM, vol.49, no.1, January 2006.

[10] G. Singh, *"PIM for Mobility"*, ACM SIGIR Workshop on Personal Information Management, pp. 98-101, Seattle, WA, August 2006.

[11] L.D. Paulson, *"Search Technology Goes Mobile"*, Computer Magazine, pp. 19-22, August 2005.

[12] R. Schusteritsch, S. Rao, and K. Rodden, *"Mobile Search with Text Messages: Designing the User Experience for Google SMS"*, pp. 1777-1780, CHI 2005.

[13] G. Singh, *"A Taxonomy of Personal Information Management for Mobile Users"*, Working paper, Department of Computer Science, Naval Postgraduate School, 2007.

[14] S. Guthery and M. Cronin, *"Mobile Application Development with SMS and the SIM Toolkit"*, McGraw Hill, 2002.

[15] Ng. YuLoon, *"Short Message Service (SMS) Security Solution for Mobile Devices"*, Master's Thesis, Naval Postgraduate School, December 2006.

[16] The Official Bluetooth Technology Info Site, *"How Bluetooth Technology Works"*, http://www.bluetooth.com/Bluetooth/Learn/Works/, last accessed 15 August 2007.

[17] D. Amit, *"Wireless Home Networks – DECT, Bluetooth, HomeRF, and Wireless LANs"*, XILINX, White Paper, 21 March 2001.

[18] S. Keshav, *"Why Cell Phones Will Dominate the Future Internet"*, ACM SIGCOMM, vol.35, no.2, April 2005.

[19] J. Wayne and R. Ayers, *"Guidelines on Cell Phone Forensics"*, National Institute of Standards and Technology, Special Publication, pp. 800-101, May 2007.

[20] R. Mossesgeld, *"What exactly is a smartphone, and what exactly is a PDA?"*, http://www.thesmartpda.com/50226711/what_exactly_is_a_smartphone_and_what_exactly_is_a_pda.php, last accessed 16 August 2007.

[21] J. Waycott and A. Kukulska-Hulme, *"Students' Experiences With PDAs for Reading Course Materials"*, 01 March 2002.

[22] M. Akbas and G. Singh, *"Personal Information Search on Mobile Devices"*, September 2007, (to appear).

[23] Y.N. Singh, *"Mobile Computing Networks"*, EE/ACES, IIT Kanpur, Seminar Slides, August 2005.

[24] E. Signorini, *"Notebook Computers Go Truly Mobile at the Intersection of 3G and IT"*, Yankee Group Consulting Report, February 2007.

[25] V. Upkar and R. Vetter, *"Emerging Wireless and Mobile Networks"*, Communications of The ACM, vol. 43, no. 6, June 2000.

[26] G. Kioumourtzis, *"Simulation and Evaluation of Routing Protocols for Mobile Ad Hoc Networks (MANETs)"*, Master's Thesis, Naval Postgraduate School, September 2005.

[27] Infrared Data Association (IrDA) Official Web Site, *"About IrDA"*, http://irda.org/displaycommon.cfm?an=1 last accessed on 17 August 2007.

[28] C. Evers, N. Akalugwu, J. Bugnevicius, and F. Pourtaran, *"General Computer Science II Course Project: Bluetooth and Infrared"*, International University Bremen, 11 April 2003.

[29] J. Brooks, *"Are Devices Better Dead Than Infrared?"*, eWEEK, January 2001.

[30] S. Keshav, *"Why Cell Phones Will Dominate the Future Internet"*, ACM SIGCOMM, Computer Communication Review, vol. 35, no.2, April 2005.

[31] C. Bettstetter, H.J. Vogel, and J. Eberspeher, *"GSM Phase 2+ General Packet Radio Service GPRS: Architecture*, Protocols, and Air Interface", IEEE Communication Survey, 1999.

[32] B.T. Schilit, D.J. Hilbert, and T.K. Koh, *"Web Interaction Using Very Small Internet Devices"*, Computer; IEEE, pp. 37-45, October 2002.

[33] M.B. Jones, G. Buchanan, and H. Thimbleby, *"Sorting out Searching on Small Screen Devices"*, Conference on Mobile HCI, September 2002.

[34] K. Church, B. Smyth, P. Cotter, and K. Bradley, *"Mobile Information Access: A Study of Emerging Search Behavior on the Mobile Internet"*, ACM Transactions on the Web, vol.1, no.1, article 4, May 2007.

[35] R. Schusteritsch, S. Rao, and K. Rodden, *"Mobile Search with Text Messages: Designing the User Experience for Google SMS"*, pp. 1777-1780, CHI 2005.

[36] J.L. Leidner, *"A Wireless Natural Language Search Engine"*, p. 677, ACM SIGIR 2005.

[37] Y. Raivio, *"Peer-to-Peer Overlay Architecture for Mobile Networks"*, pp. 1-10, HIIT 2005.

[38] C. Lindemann and O.P. Waldhorst, *"A Distributed Search Service for Peer-to-Peer File Sharing in Mobile Applications"*, Proc 2nd Int'l Conf on Peer-to-Peer Computing (P2P'02), 2002.

[39] K. Church, B. Smyth, and M.T. Keane, *"Evaluating Interfaces for Intelligent Mobile Search"*, W4A at WWW2006, 23-26 May 2006.

[40] S. Jones, M. Jones, and S. Deo, *"Using Keyphrases as Search Result Surrogates on Small Screen Devices"*, International Journal of Personal and Ubiquitous Computing, 8(1):55–68, 2004.

[41] G. Singh, *"Wireless Data Services"*, Course Slides for CS4137, Department of Computer Science, Naval Postgraduate School, March 2007.

[42] G. Singh, (ed) *"Content Repurposing"*, IEEE Multimedia, 11(1), January-March 2004.

[43] Java Platform, Micro Edition, http://java.sun.com/javame/index.jsp (accessed on 30 July 2007).

[44] Google Desktop, http://desktop.google.com/features.html (accessed on 30 July 2007).

[45] Java Community Process Program, *"JSR 75: PDA Optional Packages for the J2METM Platform"*, http://www.jcp.org/en/jsr/detail?id=75 (accessed on 18 August 2007).

[46] Java GDS API v2, *"The Apache Software Foundation"*, http://gdapi.sourceforge.net/, (accessed on 15 July 2007).

[47] D. Johnson, *“RSS and Atom in action”*, 209 Bruce Park Ave., Greenwich, CT 06830: Manning Publications Co., 2006.

THIS PAGE INTENTIONALLY LEFT BLANK

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California