MOBILE APPLICATIONS DEVELOPMENT WITH
ANDROID
Technologies and Algorithms

Meikang Qiu, Wenyun Dai, and Keke Gai
Pace University, New York City, New York, USA

© 2017 by Taylor & Francis Group, LLC
Contents

List of Figures xi
List of Tables xxi
Preface xxiii
Acknowledgments xxv
Author Biographies xxvii

SECTION 1 Mobile App Design and Development Skills

CHAPTER 1 • Overview of Mobile App and Mobile Interface 3

1.1 MOBILE SYSTEM 4
1.2 MOBILE INTERFACE AND APPLICATIONS 6
  1.2.1 Optimizations in Mobile Systems 8
  1.2.2 Mobile Embedded System 8
1.3 MOBILE CLOUD 9
  1.3.1 Big Data Application in Mobile Systems 10
  1.3.2 Data Security and Privacy Protection in Mobile Systems 11
  1.3.3 Concept of Mobile Apps 13
  1.3.4 Brief Introduction of Android and Its Framework 14
    1.3.4.1 A Brief History of Android 14
    1.3.4.2 Android Device Distribution 16
CHAPTER 4  2-D Graphics and Multimedia in Android  

4.1  INTRODUCTION OF 2-D GRAPHICS TECHNIQUES  
  4.1.1  Color  
  4.1.2  Paint  
  4.1.3  Path  
  4.1.4  Canvas  
  4.1.5  Drawable  
  4.1.6  Button Selector  

4.2  ADVANCED UI DESIGN  
  4.2.1  Multiple Screens  
  4.2.2  Action Bar  
  4.2.3  Custom Views  

4.3  OVERVIEW OF MULTIMEDIA IN ANDROID  
  4.3.1  Understanding the MediaPlayer Class  
  4.3.2  Life Cycle of the MediaPlayer State  

4.4  AUDIO IMPLEMENTATIONS IN ANDROID  

4.5  EXECUTING VIDEO IN ANDROID  

4.6  EXERCISES  
  4.6.1  Basic Exercises  
  4.6.2  Advanced Exercises  

SECTION II  Advanced Mobile App Optimizations  

CHAPTER 5  Mobile Embedded System Architecture  

5.1  EMBEDDED SYSTEMS  
  5.1.1  Embedded Systems Overview  

5.2  SCHEDULING ALGORITHMS  
  5.2.1  Basic Concepts  
  5.2.2  First-Come, First-Served Scheduling Algorithm  
  5.2.3  Shorted-Job-First Scheduling Algorithm  
  5.2.4  Multiprocessors
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.5 Priority Scheduling Algorithm</td>
<td>88</td>
</tr>
<tr>
<td>5.2.6 ASAP and ALAP Scheduling Algorithm</td>
<td>91</td>
</tr>
<tr>
<td>5.2.6.1 ASAP</td>
<td>93</td>
</tr>
<tr>
<td>5.2.6.2 ALAP</td>
<td>94</td>
</tr>
<tr>
<td>5.3 MEMORY TECHNOLOGY</td>
<td>96</td>
</tr>
<tr>
<td>5.4 MOBILE EMBEDDED SYSTEMS</td>
<td>97</td>
</tr>
<tr>
<td>5.4.1 Embedded Systems in Mobile Devices</td>
<td>97</td>
</tr>
<tr>
<td>5.4.2 Embedded Systems in Android</td>
<td>98</td>
</tr>
<tr>
<td>5.4.3 Power Management of Android</td>
<td>100</td>
</tr>
<tr>
<td>5.4.4 Embedded Systems in Mobile Apps</td>
<td>101</td>
</tr>
<tr>
<td>5.5 MESSAGING AND COMMUNICATION MECHANISMS</td>
<td>102</td>
</tr>
<tr>
<td>5.5.1 Message Mechanisms</td>
<td>103</td>
</tr>
<tr>
<td>5.5.2 Communication Mechanisms</td>
<td>104</td>
</tr>
<tr>
<td>5.6 EXERCISES</td>
<td>106</td>
</tr>
</tbody>
</table>

### CHAPTER 6 - Data Storage and SQLite Operations

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 LOCAL DATA</td>
<td>110</td>
</tr>
<tr>
<td>6.1.1 Internal and External Storage</td>
<td>110</td>
</tr>
<tr>
<td>6.1.2 Save a File on Internal Storage</td>
<td>111</td>
</tr>
<tr>
<td>6.1.3 Save a File on External Storage</td>
<td>113</td>
</tr>
<tr>
<td>6.1.4 Delete a File</td>
<td>116</td>
</tr>
<tr>
<td>6.1.5 Query the Space</td>
<td>117</td>
</tr>
<tr>
<td>6.2 SQLITE DATABASE</td>
<td>118</td>
</tr>
<tr>
<td>6.2.1 Table Structure</td>
<td>119</td>
</tr>
<tr>
<td>6.2.2 CRUD Operations</td>
<td>119</td>
</tr>
<tr>
<td>6.2.3 Usage of SQLite Techniques</td>
<td>124</td>
</tr>
<tr>
<td>6.3 CONTENT PROVIDER</td>
<td>126</td>
</tr>
<tr>
<td>6.4 EXERCISES</td>
<td>128</td>
</tr>
<tr>
<td>6.4.1 Basic Exercises</td>
<td>128</td>
</tr>
<tr>
<td>6.4.2 Advanced Exercises</td>
<td>129</td>
</tr>
</tbody>
</table>
CHAPTER 7 • Mobile Optimization by Dynamic Programming

7.1 INTRODUCTION OF HETEROGENEOUS EMBEDDED SYSTEMS AND DYNAMIC PROGRAMMING 133

7.2 FIXED TIME MODEL 134
  7.2.1 Heterogeneous Assignment 134
  7.2.2 Minimizing Costs by Scheduling 136

7.3 PROBABILISTIC TIME MODEL 143
  7.3.1 Introduction of Probabilistic Time Model 143
  7.3.2 Solutions to Heterogeneous Assignment Problems 147
  7.3.3 Generating a D Table 147
  7.3.4 Example of Generating a D Table 149

7.4 NONDETERMINISTIC POLYNOMIAL-TIME PROBLEMS 161
  7.4.1 Cook’s Theorem 161

7.5 EXERCISES 162
  7.5.1 Fundamental Questions 162
  7.5.2 Practical Questions 163

7.6 GLOSSARY 165

CHAPTER 8 • Mobile Optimizations by Loop Scheduling 167

8.1 INTRODUCTION 168

8.2 BASIC GRAPH MODELS AND TECHNIQUES 169
  8.2.1 Data-Flow Graph in Loop Scheduling 169
  8.2.2 Retiming and Unfolding 171

8.3 FUNDAMENTAL TIMING OPTIMIZATIONS 173

8.4 TIME AND POWER OPTIMIZATIONS WITH LOOP SCHEDULING 177
  8.4.1 Probabilistic Data-Flow Graph 177
  8.4.2 Loop Scheduling and Parallel Computing 180

8.5 CONCLUSIONS 186

8.6 EXERCISES 187
### Contents

- 8.6.1 Fundamental Questions 187
- 8.6.2 Practical Questions 188
- 8.7 GLOSSARY 190

**SECTION III** Mobile App Techniques in Emerging Technologies

**CHAPTER 9** Mobile Cloud Computing in Mobile Applications Deployment 195

- 9.1 INTRODUCTION 196
- 9.2 CONCEPTS OF MOBILE CLOUD COMPUTING 197
  - 9.2.1 Technological Structure of Mobile Cloud Computing 197
  - 9.2.2 Differences between Cloud Computing and Mobile Cloud 198
  - 9.2.3 Mobile Computing 200
  - 9.2.4 Wireless Networks 201
    - 9.2.4.1 Wireless Local Area Networks (WLANs) 202
    - 9.2.4.2 Comparison between Wireless Local Area Networks and Wireless Fidelity 205
    - 9.2.4.3 Wireless Wide Area Network 205
    - 9.2.4.4 Cellular Network 206
- 9.3 MAIN TECHNIQUES OF MOBILE CLOUD COMPUTING 208
  - 9.3.1 Virtualization 208
  - 9.3.2 Parallel Programming Model 210
  - 9.3.3 Mass Distributed Storage 211
- 9.4 MOBILE CLOUD COMPUTING ARCHITECTURE 212
- 9.5 EXERCISES 214
  - 9.5.1 Fundamental Questions 214
  - 9.5.2 Practical Questions 215
- 9.6 GLOSSARY 215
CHAPTER 10 • Efficient Data Synchronization on Mobile Devices in Big Data

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>OVERVIEW OF BIG DATA</td>
<td>222</td>
</tr>
<tr>
<td></td>
<td>10.1.1 Understanding Data Type</td>
<td>222</td>
</tr>
<tr>
<td></td>
<td>10.1.2 Categorizing Big Data Models</td>
<td>223</td>
</tr>
<tr>
<td></td>
<td>10.1.3 Current Challenges in Big Data</td>
<td>224</td>
</tr>
<tr>
<td>10.2</td>
<td>BIG DATA PROCESSING</td>
<td>225</td>
</tr>
<tr>
<td></td>
<td>10.2.1 Machine Learning</td>
<td>227</td>
</tr>
<tr>
<td></td>
<td>10.2.1.1 Supervised Learning</td>
<td>227</td>
</tr>
<tr>
<td></td>
<td>10.2.1.2 Unsupervised Learning</td>
<td>231</td>
</tr>
<tr>
<td></td>
<td>10.2.1.3 Semisupervised Learning</td>
<td>233</td>
</tr>
<tr>
<td>10.3</td>
<td>MOBILE BIG DATA STORAGE</td>
<td>233</td>
</tr>
<tr>
<td></td>
<td>10.3.1 Introduction and Basic Concepts</td>
<td>233</td>
</tr>
<tr>
<td></td>
<td>10.3.2 Heterogeneous Memory Architecture</td>
<td>235</td>
</tr>
<tr>
<td></td>
<td>10.3.3 Multi-dimensional Dynamic Programming Data</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>Allocation</td>
<td></td>
</tr>
<tr>
<td>10.4</td>
<td>SECURITY AND PRIVACY ISSUES</td>
<td>246</td>
</tr>
<tr>
<td>10.5</td>
<td>DATA DEDUPLICATION</td>
<td>250</td>
</tr>
<tr>
<td>10.6</td>
<td>EXERCISES</td>
<td>251</td>
</tr>
<tr>
<td></td>
<td>10.6.1 Fundamental Questions</td>
<td>251</td>
</tr>
<tr>
<td></td>
<td>10.6.2 Practical Questions</td>
<td>252</td>
</tr>
</tbody>
</table>

SECTION IV Appendixes

Appendix 1: Project 1

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROJECT DESCRIPTION</td>
<td>255</td>
</tr>
<tr>
<td>PROJECT OBJECTIVE</td>
<td>255</td>
</tr>
<tr>
<td>PROJECT GUIDELINES</td>
<td>255</td>
</tr>
<tr>
<td>PROJECT PLAN</td>
<td>256</td>
</tr>
</tbody>
</table>

Appendix 2: Project 2

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>257</td>
</tr>
</tbody>
</table>
## Contents

- **PROJECT DESCRIPTION** 257
- **PROJECT OBJECTIVE** 257
- **PROJECT REQUIREMENTS** 257
- **PROJECT PLAN** 257

**Appendix 3: BirthProvider.java. Codes** 261

- **BIRTHPROVIDER.JAVA. CODES** 261

**Bibliography** 269
Preface

Mobile applications have dramatically penetrated in numerous fields and have changed people’s lives. Developing an effective mobile app has become a significant issue for current enterprises to spread out their services or produces, and build up a direct connection with customers. A mobile application is an option of receiving the full attention of its users, which distinguishes from a desktop that has multitask offerings. The number of mobile app downloads and executions have been rising considerably over years with the rapid development of the mobile technologies. Differentiating from other mobile apps has become an important issue for enterprises to increase the mobile applications’ value. Therefore, finding an effective way to maximize the performance of mobile apps creates an urgent demand for contemporary mobile app practitioners. Using advanced techniques of mobile applications is considered an effective approach to making mobile apps stand out in a group of app selections.

This book focuses on introducing advanced techniques of mobile apps and attempts to instruct learners in skills of using those advanced approaches in practical mobile apps developments. The approaches involved in this book address the recent main achievements of mobile technologies and wireless networks. Learners can gain knowledge on a broad scope of mobile apps within the Android framework. The instructional aim is to successfully disseminate novel mobile apps development methods and enable knowledge discoveries in the field. Students will obtain updated mobile techniques and app development skills matching the graduate level studies after understanding the contents of this book. The instructions will cover a few fields, including advanced algorithms, embedded systems, novel mobile app architecture, and mobile cloud-computing paradigms.

There are mainly two concentrations in this book, namely mobile apps development and algorithms. These two concentrations are covered by three sections, which represent three major dimensions in the current mobile app development domain. They are:
1. Mobile app design and development skills. This includes Chapters 2 to 4. In Chapter 2, we offer a quick start on Android from introducing Java to running an Android application on a real phone. Chapter 3 provides an overview of key concepts and design in Android mobile applications. Finally, Chapter 4 introduces skills of 2D graphics and UI design as well as multimedia in Android mobile apps.

2. Advanced mobile app optimizations. Chapters 5 to 8 focus on this aspect. We provide an overview of mobile embedded system and its architecture in Chapter 5. Chapter 6 introduces techniques of data storage in Android. Moreover, we discuss the knowledge of mobile optimizations by dynamic programming in Chapter 7. Finally, a presentation of mobile optimizations by loop scheduling is given in Chapter 8.

3. Mobile app techniques in emerging technologies. This is discussed in Chapter 9 and Chapter 10. In Chapter 9, we discuss the techniques of mobile cloud computing in mobile applications deployment. In addition, we offer the other advanced techniques used in big data in Chapter 10, which focuses on mobile big data storage.

The overall educational objective is to enable readers to cognize the approaches of developing Android mobile apps using advanced techniques for achieving enhanced performances. The emerging technologies impacting on mobile apps are also considered. The instructional concentration targets facilitating students’ awareness of the knowledge by combining knowledge representations with practical exercises. This book’s content has been evaluated by practical instructions in higher education, such as Columbia University (New York) and Pace University (New York).
Overview of Mobile App
and Mobile Interface

Mobile system and mobile app are two fundamental aspects in Android mobile app development. In this chapter, we introduce the overview of mobile system and mobile app, which include:

1. Introduction of the mobile system.
2. Mobile interface and applications in mobile system.
3. Optimization in mobile system.
4. Mobile embedded system.
5. Mobile cloud computing.
Mobile Applications Development with Android

6. Big data in mobile system.
7. Data security and privacy protection in mobile system.
8. Mobile app.

1.1 MOBILE SYSTEM

Mobile system includes mobile device, mobile operating system, wireless network, mobile app, and app platform.

The mobile device consists of not only smartphones but also other handheld computers, such as a tablet and Personal Digital Assistant (PDA). A mobile device has a mobile operating system and can run various types of apps. The most important parts of a mobile device are Central Processing Unit (CPU), memory, and storage, which are similar to a desktop but perform weaker than an on-premise device. Most mobile devices can also be equipped with Wi-Fi, Bluetooth, and Global Positioning System (GPS) capabilities, and they can connect to the Internet, other Bluetooth-capable device and the satellite navigation system. Meanwhile, a mobile device can be equipped with some human-computer interaction capabilities, such as camera, microphone, audio systems, and some sensors.

All kinds of mobile devices run on various mobile Operating Systems (OS), also referred to mobile OSs, such as iOS from Apple Inc., Android from Google Inc., Windows Phone from Microsoft, Blackberry from BlackBerry, Firefox OS from Mozilla, and Sailfish OS from Jolla. Mobile devices actually run two mobile operating systems. Besides the mobile operating systems that end users can see, mobile devices also run a small operating system that manages everything related to the radio. Because of the high time dependence, the system is a low-level proprietary real-time operating system. However, this low-level system is security vulnerable if some malicious base station gains high levels of control over the mobile device [1, 2, 3]. We will discuss the security problem in mobile device later.

Mobile devices can connect to the Internet by wireless networks [4, 5]. There are two popular wireless networks for mobile devices: cellular network and Wi-Fi. The cellular network is peculiar to portable transceivers. A cellular network is served by at least one fixed-location
transceiver, called cell site or base station, as shown in Fig. 1.1. Each mobile device uses a different set of frequencies from neighboring ones, which means a mobile device must connect to the base station before it accesses to the Internet [6]. Similarly, when a mobile device using a cellular network wants to connect another mobile device, it must connect to some base stations before it communicates with the target device via the base stations.

![Figure 1.1 Structure of a cellular network.](image)

Wi-Fi is a local area wireless technology, which allows mobile devices to participate in computer networks using 2.4 GHz and 5 GHz radio bands. Fig. 1.2 represents two common logos of Wi-Fi. Mobile devices can connect to the Internet via a wireless networking access point. The valid range of an access point is limited, and the signal intensity descends as the distance increases. Wi-Fi allows cheaper deployment of Local Area Networks (LAN), especially for spaces where cables cannot be run. Wi-Fi Protected Access encryption (WPA2) is considered a secure approach by providing a strong passphrase. A Wi-Fi signal occupies five channels in the 2.4 GHz band. Any two channel numbers differ by five or more. Many newer consumer devices support...
the latest 802.11ac\(^2\) standard, which uses the 5 GHz and is capable of multistation WLAN throughput of at least 1 gigabit per second.

1. Hz is the unit of frequency in the International System of Units and is defined as one cycle per second. One gigahertz (GHz) represents \(10^9\) Hz.

2. IEEE 802.11ac was approved in January 2014 by IEEE Standards Association.

A mobile app is a program designed to run on smartphones, tablet computers, and other mobile devices. Mobile apps emerged in 2008 and are operated by the owner of the mobile operating systems. Currently, the most popular digital distribution platforms for mobile apps are App Store, Google Play, Windows Phone Store, and BlackBerry App World, as shown in Fig. 1.3. These platforms are developed by Apple Inc., Google, Microsoft, and BlackBerry Ltd., respectively, and provide different apps, which only can be used on their own operating systems.

Figure 1.3 Four dominate platforms for mobile apps.

1.2 MOBILE INTERFACE AND APPLICATIONS

Mobile devices, to some extent, are much more powerful than desktops. They are highly personal, always on, always with users, usually connected, and directly addressable. Furthermore, they are crawling with powerful sensors with various functions that detect location, acceleration, orientation, movement, proximity, and surrounding conditions. The portability of mobile devices combined with powerful sensors makes mobile interface extremely valuable for using mobile devices.

The User Interface (UI) is the look and feel of the on-screen system, including how it works, its color scheme, and how it responds to users’ operation. The interactions include not only users’ active operations, but also the passive ones. Users’ passive operations include users’ locations, movements, and other information that does not need users’ active operations. We will take telehealth as an example of mo-
Mobile interface. Telehealth is the delivery of health-related services and information via telecommunications technologies [7].

We can separate telehealth system into several modes: store-and-forward, real-time, remote patient monitoring, and electronic consultation, as shown in Fig. 1.4. Each mode finish their job respectively and achieve the whole process of collecting data from users, transmitting this data to medical or clinical organizations, medical reasoning and decision, and sending back to users. In the first step, observations of daily living and clinical data are captured and stored on the mobile device. All the sensors that collect and record data are heterogeneous medical devices with different cost and time features. Then the mobile device transmits this information to the Telehealth pervasive computing platform and cloud platform by wireless network [8, 9].

Consequently, main challenges include finding out the approach of collecting data from users by using sensors and scheduling sensors for achieving energy-aware purposes [10, 11]. The process of transmitting data is a part of real-time system. Different to normal real-time systems, the data transmitting in telehealth is under a wireless condition. Similar to the first step, there are various network paths with different cost and time requirements, which results in a great challenge to security and data integrity [12].
Furthermore, context-aware medical reasoning and decision is another important issue in telehealth system. Context can refer to real world characteristics, such as temperature, time or location. Combining with users’ personal information, the medical reasoning and decision focus on data analytic, mining, and profiling issues. In conclusion, all the challenges mentioned above can be summarized as a general problem: how to minimize the total cost of heterogeneous telehealth while finishing the whole diagnosis within certain time constraints [13, 14].

1.2.1 Optimizations in Mobile Systems

All current mobile devices are battery-powered devices. The high usage of mobile devices makes them hard to keep on charging like desktops, so the improvement of battery life on mobile devices is gaining increasing attention. Besides some energy-saving operations by users, there are some researches focusing on the optimization in mobile system [15, 16]. The optimization problem, to some extent, is a tradeoff among multiple constraints [17, 18]. Before talking about the optimization, let us discuss some constraints in mobile systems.

The first and the most important constraint is the energy. The second one is the performance. The third one is the networking speed to the Internet. The fourth one is the resources of the mobile device [19]. These constraints are interrelated and mutually restrict to each other. Suppose in an extreme situation, someone keeps his/her mobile device off. In this situation, the battery life can last an almost unlimited time without considering the self-discharge of the battery. However, the mobile device in that situation is useless, and no one buys a mobile device just for decoration. It is obvious that the more functions users use, the more energy devices consume. Similarly, the performance is related to the networking speed while constrained by the energy and resource. To solve this problem, many researchers proposed various optimization algorithms and frameworks [20, 21, 22].

1.2.2 Mobile Embedded System

An embedded system is a computer system with a dedicated function, which is embedded as a part of a complete devices including hardware and mechanical parts. Embedded systems are driving an information revolution with their pervasion. These tiny systems can be found everywhere, ranging from commercial electronics, such as cell phones,
cameras, portable health monitoring systems, automobile controllers, robots, and smart security devices, to critical infrastructure, such as telecommunication networks, electrical power grids, financial institutions, and nuclear plants [23, 24]. The increasingly complicated embedded systems require extensive design automation and optimization tools. Architectural-level synthesis with code generation is an essential stage toward generating an embedded system satisfying stringent requirements, such as time, area, reliability, and power consumption, while keeping the product cost low and development cycle short.

A mobile device is a typical embedded system, which includes mobile processors, storage, memory, graphics, sensors, camera, battery, and other chips for various functions. The mobile device is a high-level synthesis for real-time embedded systems using heterogeneous functional units (FUs) [20, 22]. A functional unit is a part of an embedded system, and it performs the operations and calculations for tasks. As a result, it is critical to select the best FU type for various tasks.

1.3 MOBILE CLOUD

Limited resources is another critical characteristic of mobile devices [25, 26]. With the development of cloud computing, mobile cloud computing has been introduced to the public [27]. Mobile cloud computing, as shown in Fig. 1.5, is the combination of cloud computing, mobile computing, and wireless networks to bring rich computational resources to the mobile system. In general, a mobile device with limited resources can utilize computational resources of various cloud resources to enhance the computational ability of itself. There are several challenges in mobile cloud computing, such as moving computational processes from mobile devices to the cloud, networking latency [28], context processing, energy management [29, 30], security [31], and privacy [32].

![Diagram](image.png)

Figure 1.5 Main structure of mobile cloud computing.
Currently, some research and development addresses execution code offloading, seamless connectivity and networking latency; however, efforts still lack in other domains.

**Architecture.** The architecture for a heterogeneous mobile cloud computing environment is crucial for unleashing the power of mobile computing toward unrestricted ubiquitous computing.

**Energy-aware transmission.** Offloading executive codes into the cloud can greatly reduce the burden and the time of local mobile devices, but increase the transmission between mobile devices and the cloud. The transmission protocol should be carefully designed for saving energy.

**Context-aware computing.** Context-aware and socially aware computing are inseparable traits of mobile devices. How to achieve the vision of mobile computing among heterogeneous converged networks among mobile devices is an essential need [33].

**Live Virtual Machine (VM) migration.** A virtual machine is an emulation of a particular computer system. Executive resource offloading involves encapsulation of a mobile app in a VM instance, and migrating in the cloud is a challenging task.

**Security and privacy.** Due to lack of confidence in the cloud, many users are concerned with the security and privacy of their information. It is extremely important to improve the security and the privacy of mobile cloud computing.

### 1.3.1 Big Data Application in Mobile Systems

Big data is an all-encompassing term for any collection of data sets so large or complex that it becomes difficult to process them using traditional data processing applications. Data sets grow in size in part because they are increasingly being gathered by mobile devices. There are 4.6 billion mobile phone subscriptions worldwide and between 1 billion and 2 billion people accessing the Internet [34].

With billions of mobile devices in the world today, mobile computing is becoming the universal computational platform of the world [35]. These mobile devices generate huge amounts of data every day. The
rise of big data demands that we be able to access data resources anytime and anywhere about every daily thing. Furthermore, these kinds of data are invaluable and profitable if used well.

However, a few challenges must be addressed to make big data analytics possible. More specifically, instead of being restricted to single computers, ubiquitous applications must be able to execute on an ecosystem of networked devices, each of which may join or leave the shared ubiquitous space at any time. Moreover, there exist analytics tasks that are too computationally expensive to be performed on a mobile device ecosystem. Also, how can we harness the specific capabilities of each device, including varying display size, input modality, and computational resources?

1.3.2 Data Security and Privacy Protection in Mobile Systems

Due to the universality and the particularity of mobile systems to desktop system, the security in mobile systems is much more complicated and important than that in desktop systems [36, 37]. The security in mobile systems can be separated into a few parts [38].

The first threat is the malware (virus). Mobile malware is a malicious software that targets mobile devices and results in the collapse of the system and loss or leakage of information [39]. According to the June 2014 McAfee Labs Threat Report, new mobile malware has increased for five straight quarters, with a total mobile malware growth of 167 percent in the recent past years. Security threats are also growing with 200 new threats every minute [40, 41, 42]. In addition to 2.4 million new samples of mobile malware, 2013 also brought 1 million new unique samples of ransomware, 5.7 million new malicious signed binaries, and 2.2 million new Master Boot Record (MBR)-attack-related samples. The most frequent two incentives are exfiltrating user information and premium calls or SMS. Furthermore, there are some other incentives, such as sending advertisement spam, novelty and amusement, and exfiltrating user credentials [43, 44].

Another research issue is the security frameworks or approaches for detecting mobile malware [45]. There are several approaches for monitoring mobile devices and detecting mobile malware. The signature-based solution is an approach used for detecting attacks, but it fails miserably in detecting the sophisticated cyber-criminal who targets specific organizations with exploits tailored to those victims. From a process perspective, when it comes to validating a threat and subse-
quent root cause analysis, first-level responders have to send all data that looks like malicious code to the reverse engineers. This process often causes delays, because these malware teams are typically inundated.

Meanwhile, with the development of technology, an efficient representation of malware behaviors using a key observation often reveals the malicious intent even when each action alone may appear harmless [46]. The logical ordering of an application’s actions are often over time. Based on this idea, researchers present various approaches to monitor and detect malicious behavior using static analysis on data flow [47, 48, 49, 50].

Next security problem is the data over-collection behaviors in mobile apps. Current mobile phone operating systems only provide coarse-grained permissions that determine whether an app can access private information, while providing few insights into the scope of private information being used. Meanwhile, only a few users are aware of permissions information during the installations. Furthermore, some users choose to stop installing or to uninstall an app when the system warns them and asks for permission, even though they know it may bring some hidden security troubles. For example, we take location data and analyze the current status and discuss the risks caused by over collecting it.

Location data are the most frequently used data in smartphones. It can be used in apps whose main functions include maps, photo organization, shopping and restaurant recommendations, and weather. From the report of Apptivity [51], 50% of the top iOS free apps and 24% of the top iOS paid apps track a user’s location. Although users are warned whenever an app intends to capture their locations, they usually choose to allow the permission for the function offered by the app. Apps that over collect location data can be separated into two main types: location service as main function and location service as the auxiliary function. The first type of apps normally ask users for permissions to their location information, while the other app type can collect users’ location information without noticing users. The first and the most direct risk is a physical security concern. Users’ tracks are easily exposed to those who have users’ real-time and accurate location data. Users’ habits and customs are easy to be inferred by using simple data mining methods.

Furthermore, solving the data over collection problem is also a research issue in mobile apps. PiOS [47], presented by M. Egele et al., to
detect privacy leaks in iOS applications, used static analysis to detect sensitive data flow to achieve the aim of detecting privacy leaks in applications in iOS. Sharing a similar goal with PiOS, TaintDroid [48], is a system wide dynamic taint tracking multiple sources of sensitive data. The main strategy of TaintDroid is real-time analysis by leveraging Android’s virtualized execution environment [52]. Another secure model via automated validation uses commodity cloud infrastructure to emulate smartphones to dynamically track information flows and actions. This model automatically detects malicious behaviors and sensitive data misuse via further analysis of dependency graphs based on the tracked information flows and actions.

These approaches or techniques mentioned above only focus on monitoring and detecting apps. The prerequisites are that apps already gain permissions from users. However, these solutions only provide methods of monitoring and detecting behaviors of data over-collections. This approach leaves remedying operations to users, such as disabling the permissions of apps or uninstalling those apps. Users have to manually disable permissions of these apps that over collect users’ data or uninstall them. Furthermore, running these approaches or tools adds the consumption of energy, which is particularly valuable for smartphones with limited resources. As a result, the active method of avoiding data over collection behaviors in mobile apps is a crucial challenge that needs to be solved.

1.3.3 Concept of Mobile Apps

Mobile apps were originally developed to offer general productivity and information retrieval, including email, calendar, contacts, and weather information. However, with the rapid increment of public requirement, mobile apps expand into lots of other categories, such as games, music, finance, and news.

A lot of people distinguish apps from applications in a perspective of device forms. They think that applications are used on a desktop or laptop, while apps are used on a phone or tablet. Nevertheless, this simplistic view is too narrow and no longer the consensus, because apps can be used on desktops, and, conversely, applications can run on phones. At Gartner Portals, Content and Collaboration Summit 2013, many experts and developers participated a roundtable discussion titled “Why an App is not an Application” [53, 54]. They proposed that
the difference between app and application is not about the delivery mechanism and landed on a consensus that:

\[ \text{App} = \text{software designed for a single purpose and performs a single function.} \]

\[ \text{Application} = \text{software designed to perform a variety of functions.} \]

From the view of users, they don not care whether it is an app or an application by definition, and they just want to accomplish their tasks easily. Meanwhile, from the view of developers, the question they should answer is not whether they should be building an app or an application, but how they can combine the best of both into something users love.

1.3.4 Brief Introduction of Android and Its Framework

1.3.4.1 A Brief History of Android

Android was founded in Palo Alto, California, in October 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White in an effort to develop a smarter mobile device that is more aware of its owner’s location and preferences. Then to Google acquired Android Inc. and key employees, including Rubin, Miner, and White, on August 17, 2005. At Google, the team, led by Rubin, developed a mobile device platform powered by the Linux kernel. Google had lined up a series of hardware components and software partners and signaled to carriers that it was open to various degrees of cooperation on their part. On November 5, 2007, the Open Handset Alliance unveiled itself with a goal to develop open standards for mobile devices. This alliance includes technology companies, like Google, device manufacturers such as HTC, wireless carriers such as T-Mobile, and chipset makers such as Qualcomm. Then, on October 22, 2008, the first commercially available smartphone running Android came out with a fantasy name: HTC Dream. Since 2008, Android has seen numerous updates that have incrementally improved the operating system, adding new features and fixing bugs in previous releases. There are some milestones of Android SDK, such as Android SDK 2.0 (Eclair) in 2009, Android SDK 3.0 (Honeycomb) for tablets only in 2011, Android SDK 4.0 (Ice Cream Sandwich) in 2011, Android 4.1 to 4.3 (Jelly Bean) in 2012, Android SDK 4.4 (KitKat) in 2013, and Android SDK 5.0 (Lollipop) in 2014.
Figure 1.6  Android device distribution in January and July 2012.

Figure 1.7  Android device distribution in August 2012 and August 2013.

Figure 1.8  Android device distribution in January 2014 and January 2015
1.3.4.2 Android Device Distribution

Fig. 1.6 shows the Android device distributions in 2012. We can see that Android 2.3.3 and 2.2 dominate more than half of the market. Nonetheless, in the second half of 2012, Android 4.0.3 became more and more popular. In August 2013, Android 4.0 and 4.1, named Ice Cream Sandwich and Jelly Bean, respectively, surpassed Android 2.0s and dominated the Android market, as shown in Fig. 1.7. In January 2014, Android 4.1 to 4.3 still dominated the Android market. However, after one year, Android 4.4, named KitKat, rapidly occupied 39.1% of the whole market, as shown in Fig. 1.8.

1.3.4.3 Android SDK

Android SDK is open-source and widely used, which makes it the best choice for teaching and learning mobile development. Android is a software stack for mobile devices, and it includes a mobile operating system, middleware, and some key applications. As shown in Fig. 1.9, there are Linux kernel, libraries, application framework, and applications and widgets, from bottom to top. We will introduce them one by one.

The Linux kernel is used to provide some core system services, such as security, memory management, process management, power management, and hardware drivers. These services cannot be called by Android programs directly and is transparent to users. The next layer above the kernel is the native libraries, which are all written in C or C++. These libraries are compiled for the particular hardware architecture used by the mobile devices. They are responsible for handling structured data storage, graphics, audio, video, and network, which only can be called by higher-level programs. Meanwhile, Android runtime is also on top of the kernel, and it includes the Dalvik virtual machine and the core Java libraries [55].

What is Dalvik? Dalvik is the process virtual machine in Google’s Android operating system, which specifically executes applications written for Android. Programs are written in Java and compiled to bytecode for the Java virtual machine, which is then translated to Dalvik bytecode and stored in .dex and .odex files. The compact Dalvik executable format is designed for systems with limited resources.

The application framework layer provides the high-level building blocks used for creating application. It comes preinstalled with An-
Overview of Mobile App and Mobile Interface

Figure 1.9 Android system architecture.

droid, but can be extended with its own components as needed. We will introduce some basic and important building blocks of Android.

**Activity.** An activity is a user interface screen. A single activity defines a single screen with a user interface, and it defines simple life cycle methods like onCreat, onResume, and onPause for handling interruptions. Furthermore, applications can define one or more activities to handle different phases of the program.

**Intent.** An intent is a mechanism for describing a specific action, such as “pick a photo”, or “phone home”. In Android, everything goes through intents, and developer, have plenty of opportunities to replace or reuse components. Intents can be implicit or explicit. An explicit intent can be to invoke another screen when a button is pressed on the Activity in context. An implicit intent is when you create an intent and hand it off to the system to handle it.

**Service.** A service is a task that runs in the background without the user’s direct interaction. In fact, it does the majority of pro-
cessing for an application. Developers can sub-class the Service class to write their own custom service.

Content Provider. A Content provider is a set of data wrapped up in a custom Application Programming Interface (API) to read and write it. This is the best way to share global data between applications. The content provider provides a uniform singular interface to the content and data and provides a consistent interface to retrieve/store data via RESTful model supporting create, read, update, and delete (CRUD) operations.

An Android Emulator, as shown in Fig. 1.10, called Android Virtual Device (AVD), is essential to testing Android app but is not a substitute for a real device. AVDs have configurable resolutions, RAM, SD cards, skins, and other hardware. If you have installed Android SDKs, the AVD Manager can allow you to create AVDs that target any Android API level.

Figure 1.10 Android Emulator.
An Android emulator has the following basic functions:

- Host computer’s keyboard works as keyboard of device.
- Host’s mouse acts as finger.
- Connecting to the Internet using host’s Internet connection.
- Buttons: Home, Menu, Back, Search, Volume up and down.
- Ctrl-F11 toggle landscape to portrait.
- Alt-Enter toggle full-screen mode.

However, emulators have some limitations. They do not support for:

- Placing or receiving actual phone calls.
- USB and Bluetooth connections.
- Camera or video capture as input.
- Device-attached headphones.
- Determining connected state.
- Determining battery charge level and AC charging state.
- Determining SD card insert or eject. SD card is a nonvolatile memory card used extensively in portable devices.
- Simulating the accelerometer.

Then we will introduce the process of producing an Android app. In Fig. 1.11, an android app is written in Java and generates .java file. Then javac compiler .java reads source files and transforms java code into byte code. Then Dalvik takes responsibility for handling these byte codes combining with other byte codes for other .class files, and generates classes.dex. At last, classes.dex, resources, and AndroidManifest.xml cooperate and generate an .apk file, which is a runnable Android app.

Every Android app must have an AndroidManifest.xml file in its root directory. The manifest presents essential information about the application to the Android system, information the system must have
before it can run any of the application’s code. The AndroidManifest.xml file names the Java package for the application and describes the components of the application, including the activities, services, broadcast receivers, and content providers that the application is composed of. The file also names the classes that implement each of the components and publishes their capabilities. These declarations let the Android system know what the components are and under what conditions they can be launched.

Furthermore, AndroidManifest.xml file determines, which processes will host application components, and it declares which permissions the application must have in order to access protected parts of the API and interact with other application. The file also declares the permission that others are required to have in order to interact with the application’s components and lists the instrumentation classes that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is published. It declares the minimum level of the Android API that the application requires, and it lists the libraries that the application must be linked to.

In next chapter, we will discuss the mobile embedded system architecture.
1.4 EXERCISES

Basic
1. What components do a mobile systems include?
2. Does the mobile device only mean smartphones?
3. How many mobile operating systems are running on mobile devices? What are they?
4. What is the foundation of the cellular network?
5. How does a mobile device connect to the Internet under the cellular network environment?
6. What is Wi-Fi?
7. How does a mobile device connect to the Internet under the Wi-Fi network environment?
8. What is the relationship between the distance and the signal intensity under the Wi-Fi network environment?
9. Can an Android device run an iOS app?
10. What is the difference between the interface of mobile devices and that of desktops?
11. What is telehealth?
12. What is the greatest challenge in the step of collecting and transmitting data in telehealth systems?
13. What is an optimization problem? (optional question)
14. How many constraints are there in a mobile system? What are they?
15. What is an embedded system?
16. What is a mobile embedded system?
17. What is the biggest benefit of mobile cloud for mobile system?
18. What is big data in mobile system?
19. What is the malware in mobile devices?
20. What is the data over collection behavior in mobile apps?
21. What is Android?
22. What is the Open Handset Alliance?
23. When did Android 2.0 release?
24. What was the name of the first Android device?
25. Which version of Android was the most widely used in January 2014?
26. What are the main functions provided by the Linux Kernel in Android?
27. What is an Activity in Android?
28. What is an Intent?
22 ▉ Mobile Applications Development with Android

29. What is an implicit Intent?
30. What is an explicit Intent?
31. What is a Service?
32. What does a Content provider do?
33. What is the process of generating an Android app from Java code to .apk file?

Advanced
34. Why is the energy consumption more important for mobile devices than that of desktops?
35. What is the difference between apps and applications?
36. What is Dalvik? What is the difference between Dalvik and traditional Java Virtual Machine (JVM)?