COP 4600 – Summer 2014

Introduction To Operating Systems

Android Operating System

Instructor : Dr. Mark Llewellyn
markl@cs.ucf.edu
HEC 236, 407-823-2790
http://www.cs.ucf.edu/courses/cop4600/sum2014

Department of Electrical Engineering and Computer Science
Computer Science Division
University of Central Florida

Android Operating System

- Android is design to run mobile devices, specifically smart phones and tablets.
- It is built on a Linux foundation and relies on Linux to perform some of the most fundamental tasks, including management of main memory, processors, device drivers, and network access.
- The most customizable part of Android is its user interface, which can be arranges by each user to include almost any configuration of applications (apps).
- Apps are programmed in Java using a software development kit (SDK) that's available free for downloading.
- Like Linux, Android is an open source OS, publishing key components of its source code (but not all of it). While not as open as Linux, it is much more so than the OS that runs Apple's mobile devices.

A Brief History Of Android

- Andy Rubin was a co-founder of Danger Inc., and he formed the team that created the Android OS to power his company's new cell phone, called the Sidekick, which was a precursor to today's smartphones.
- The key challenge that they faced was to create a complete computing environment that could successfully manipulate the phone despite battery power limitations, a small CPU, and reduced memory space. Using Linux as the base, the team was able to create a multi-level system that integrated Linux to perform user actions via unique apps from the phone's screen.
- Google purchased Android in 2005 and Rubin joined the company. In 2013, Google CEO Larry Page announced that "more than 750 million devices had been activated globally with more than 2.5 billion apps downloaded from Google Play."

A Brief History Of Android

- As of this month (July 2014), Google Play currently lists more than 1.3 million apps available. Currently about 25,000 new apps appear every month.
- With each new release of Android, the source code is released so that manufacturers and developers can prepare their installation and customized software. You can obtain the source code at: https://source.android.com/.

A Brief History Of Android

Year	Release	Code Name	Features
2008	Version 1.0		First product available to the public
2009	Version 2.0	Éclair	Based on Linux kernel 2.6.29
2011	Version 3.0	Honeycomb	A tablet-only version based on Linux kernel 2.6.36
2011	Version 4.0	Ice Cream Sandwich	Based on Linux kernel 3.0.1
2012	Version 4.1	Jelly Bean	Based on Linux kernel 3.0.31
2013	Version 4.4	KitKat	Currently available
2014	Version L	?	Available for developers now.

Design Goals

- The goals of the Android system are focused on the user experience in a mobile environment, using a touch screen and connecting to networks through either telephony (using 3G and 4G currently) or Wi-Fi.
- The following page defines the overall design goals of Android. This was taken directly from : http://developer.android.com/design/get-started/creative-vision.html

et-started/creative-vision.html

Android - 4.4 KitKat

Creative Vision | Android De...

Race Res... News 13 - Orlando News, ... 10 Day Weather Forecast f...

Page Safety Tools

Creative Vision

Starting with Ice Cream Sandwich, we focused the design of Android around these three overarching goals, which apply to our core apps as well as the system at large. As you work with Android, consider these goals.

Enchant me

Beauty is more than skin deep. Android apps are sleek and aesthetically pleasing on multiple levels. Transitions are fast and clear; layout and typography are crisp and meaningful. App icons are works of art in their own right. Just like a well-made tool, your app should strive to combine beauty, simplicity and purpose to create a magical experience that is effortless and powerful.

Simplify my life

Android apps make life easier and are easy to understand. When people use your app for the first time, they should intuitively grasp the most important features. The design work doesn't stop at the first use, though. Android apps remove ongoing chores like file management and syncing. Simple tasks never require complex procedures, and complex tasks are tailored to the human hand and mind. People of all ages and cultures feel firmly in control, and are never overwhelmed by too many choices or irrelevant flash.

Make me amazing

It's not enough to make an app that is easy to use. Android apps empower people to try new things and to use apps in inventive new ways. Android lets people combine applications into new workflows through multitasking, notifications, and sharing across apps. At the same time, your app should feel personal, giving people access to superb technology with clarity and grace.

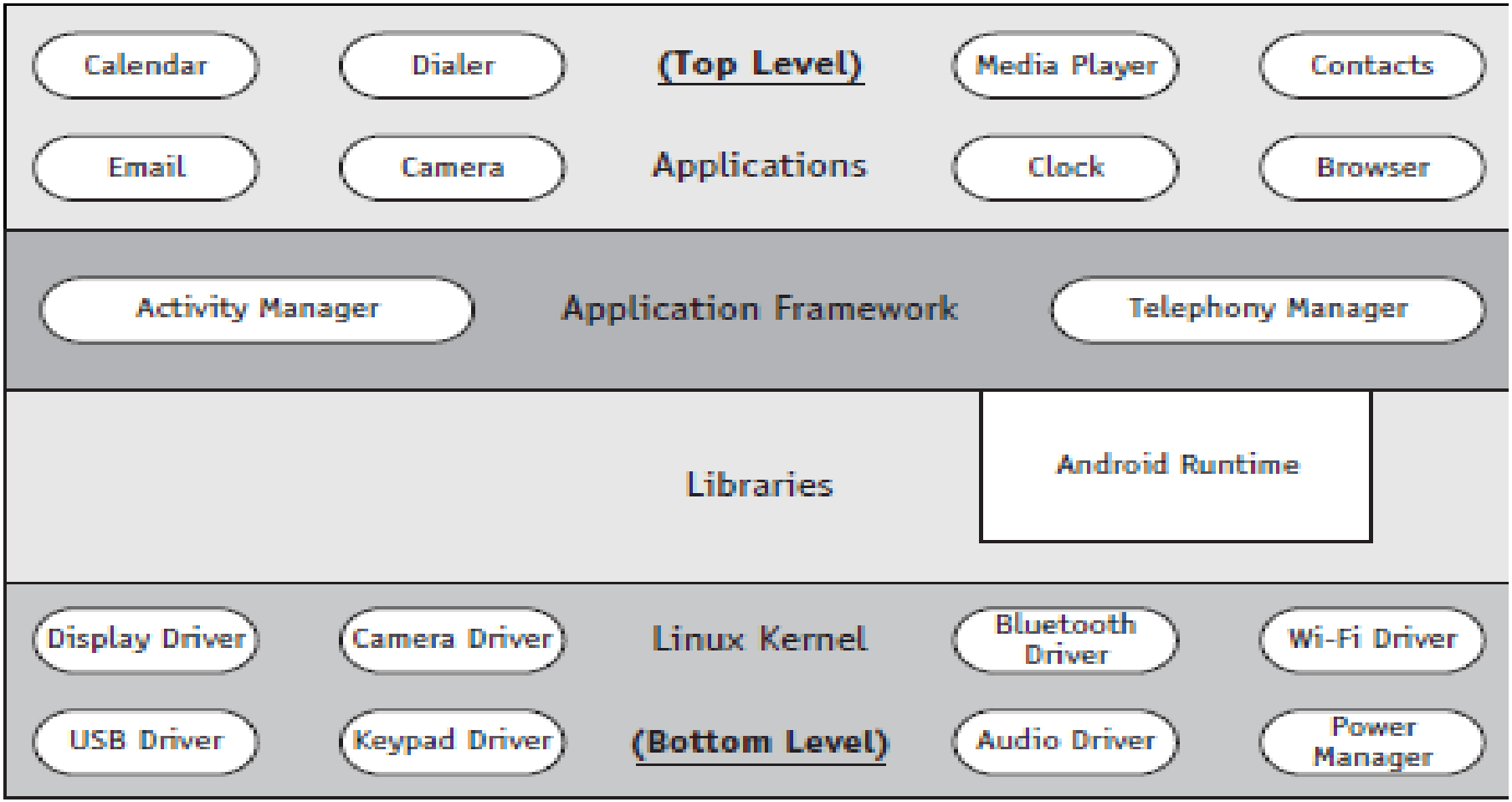
NEXT: DESIGN PRINCIPLES >

g+1 954

Memory Management

- Memory management is handled by the Linux kernel, with the help of several software modifications (including shared memory allocators) to help Android work successfully on mobile devices that are typically built with a limited amount of main memory and slower CPUs.
- Therefore, Android apps are explicitly designed to use resources only when they are needed, and to require minimal resources when they are dormant. That is, they are built to reside in memory in a sleep-like state while consuming minimal resources.
- The next page provides an illustration of that shows the Android software stack is built on a Linux kernel, which manages all device drivers (only a few illustrated here).

Memory Management



Android Software Stack

Memory Management

- Once an app is opened, it remains resident in main memory, even when it appears that it has been closed.
- By remaining in main memory, an app can usually open quicker when it is called in the near future. However, this does not mean that these open apps are not monitored.
- Android uses a LRU algorithm to keep track of each resident process and when it was most recently called. Then, if memory space should become scarce, a low memory killer (called LMK) acts to free up memory by removing the processes that have remained dormant the longest.
- In this way, users are not encouraged to manually "force stop" applications that they are not currently working with; in fact, doing so will cause these apps to take longer to open.

Processor Management

- Android processor management requires four key objects: manifest, activities, tasks, and intents.
- Manifest: a file that holds essential information that the system must have before it can run an application. The manifest includes all of the permissions that the app must have before it can begin as well as the permissions that other apps must have to work with that app's components. This information is held in a file called: `AndroidManifest.xml`
- Activity: is the application component that defines the user interface screen that the individual uses to interact with the app, including all the actions that can be performed. In general, an app has a collection of activities, including some that are unique to the app as well as activities from other cooperative apps.

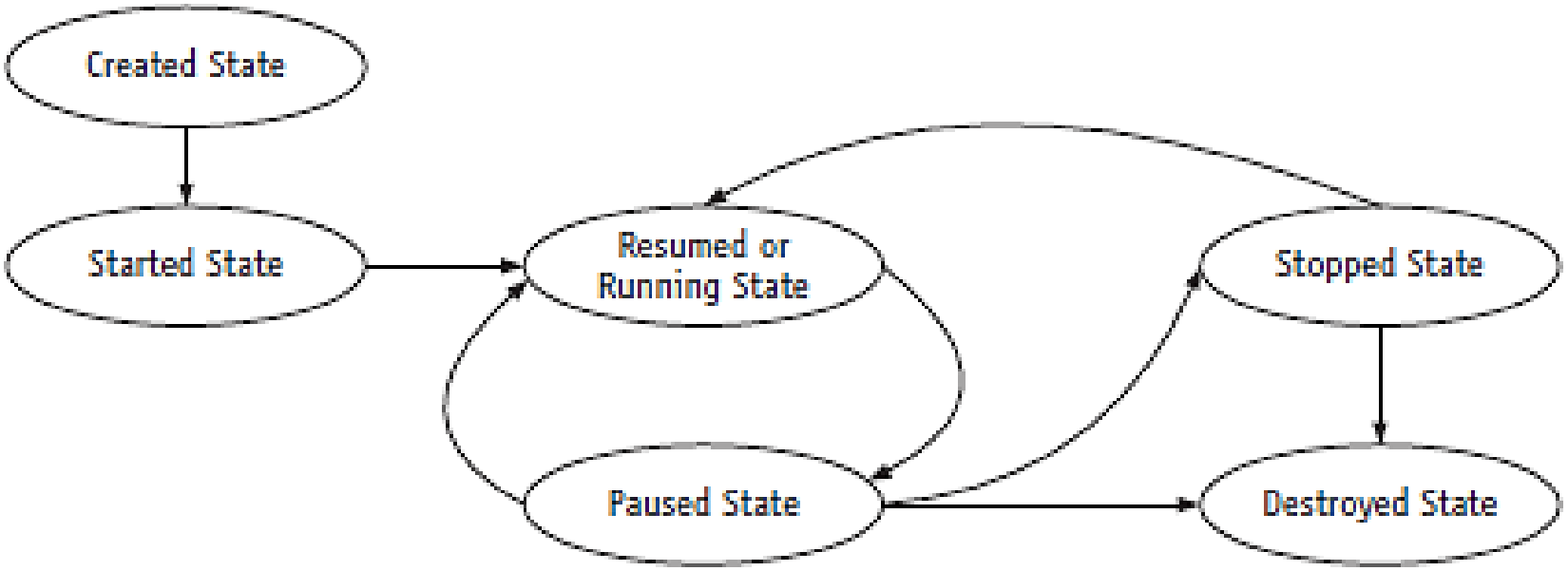
Processor Management

- Task: is defined in Android as "sequence of activities a user follows to accomplish a goal". Therefore, a task can consist of activities from just one app or from several apps. A task that runs in the background is called a service, such as a media player that continues to play even as the user moves to another app and another activity.
- Intent: is the mechanism that one app uses to signal to another app that its cooperation is requested to accomplish something. This allows apps to call on one another as needed to meet a user's request. For example, if the user requests that an app integrate a photo, that app can call on the camera app to make itself ready to take the desired picture. Once taken, the digital image will be sent to the referring app without the need to notify the user that one app is pausing while another executes, and then pausing so the first app can be resumed.

Processor Management

Activity States

- Each application can have one to many activities and each one is responsible for maintaining its processing state. This is illustrated below:



Activity States

- There are several states that an activity goes through from creation to destruction. The flow of these states is very similar to the process lifecycle we examined when dealing with process management earlier this semester.
- Created state: transient state when the activity has just begun
- Started state: software initialization begins and the first screen can be drawn. Generally considered the main screen.
- Resumed state (running state): activities execute until they are interrupted by another activity or a user command.
- Paused state: a stop for an activity that is interrupted and ready to go into a "background" mode
- Stopped state: activities disappear from the user's view. Subsequently, activity may be terminated or it may be recalled.
- Destroyed state: formal indication that the activity is terminated. It will be removed completely from system memory. Background activities closed properly to prevent memory leaks

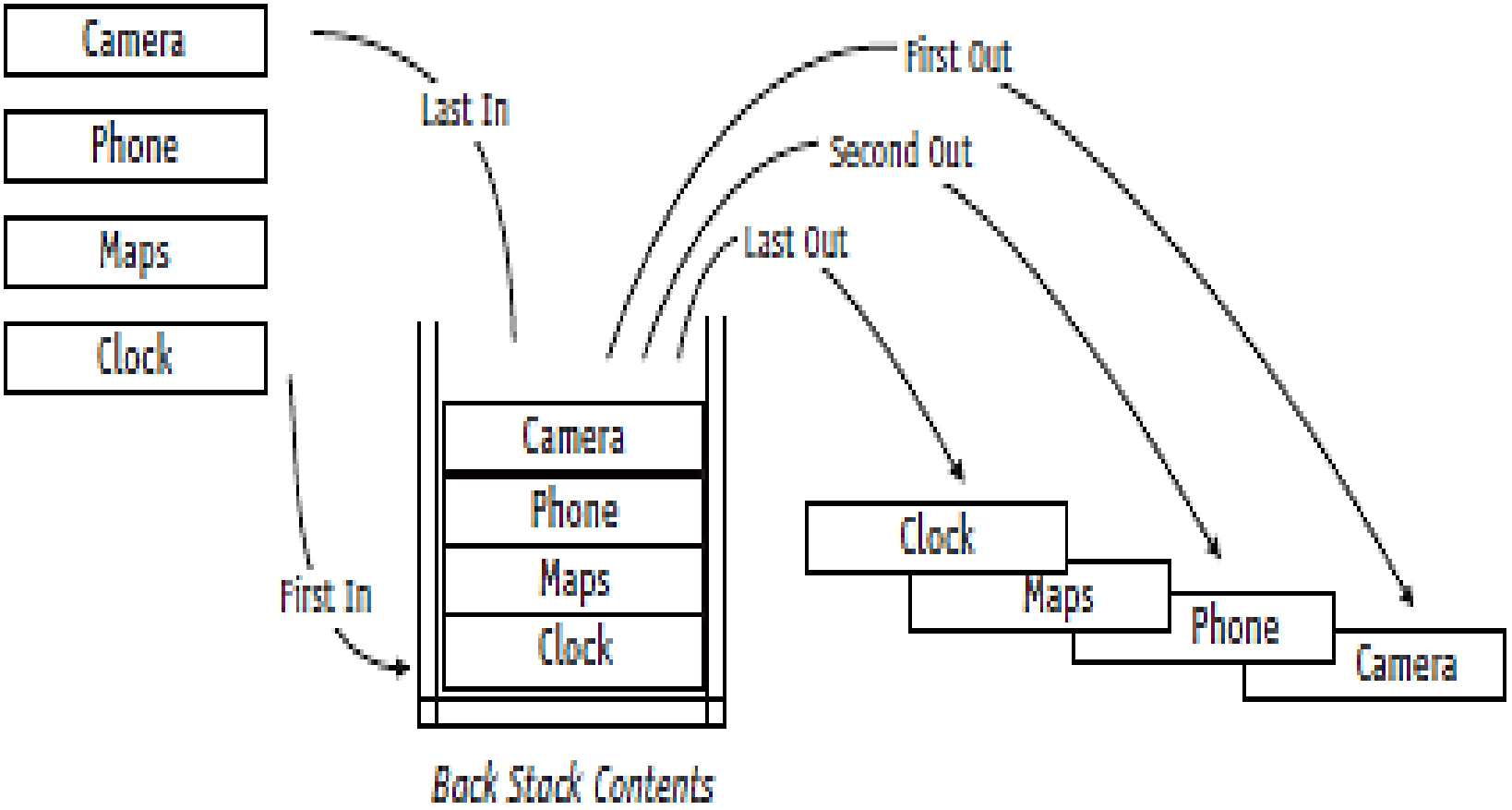
Process Management

- Whenever a new activity is begins, it replaces the activity that was previously called and its activity state is moved to a data structure called the "back stack".
- This stack is loaded and unloaded using a LIFO scheme. Each time that an app is replaced by another, such as when the user opens mail first and then opens the calendar, the status of the replaced app (mail) is moved into the stack so that the new app (calendar) can take over the screen. The contents of the stack are loaded in chronological order from the first one that was replaced to the last one replaced.
- Later, when a user wants to backtrack to the previously viewed apps and presses the "back" button on the screen, it stops the current activity (calendar) and replaces it with the screen for the most recently opened app (mail) and restores that app's status so that it appears as it was left.

Process Management

- This stacking process is shown in more detail on the next slide, which represents an example where a user opens four apps in the following order: clock (to check an appointment time), maps (to verify a location), phone (to confirm the appointment), and camera to take a picture.
- When each app is opened, it causes the status of the previously active app to be stored in the back stack. For example, when the phone app is opened, then the status of the maps app is moved to the back stack.
- After the photo is taken and the user presses the back button, the camera app disappears from the screen and the next-to-last app (the phone app) is restored with the same status (same contact and phone number) it was displaying when it was stopped.
- Pressing the back button once again causes the phone app to disappear and the maps app to replace it, with its status restored. Finally, pressing the back button again causes the maps app to disappear and the clock to open.

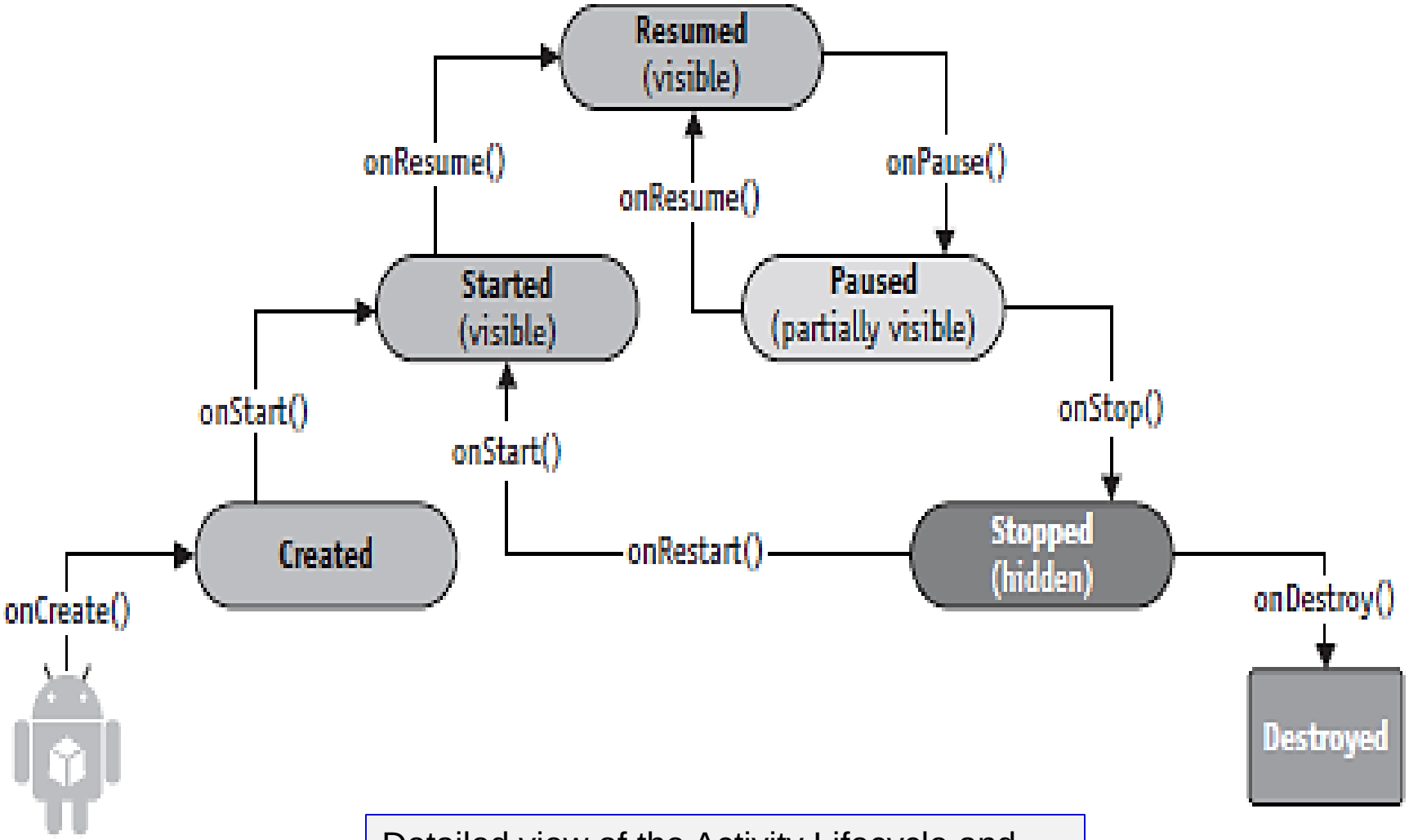
Process Management



The last-in, first-out (LIFO) scheme moves the activity status of the current app into the back stack when that app is replaced by another one. Later, it restores each app in reverse order each time the user presses the back button.

Process Management

- As activities move through their individual lifecycles, the system relies on "callbacks" for changes in states.
- Examples of these callbacks are shown on the next page.
- For example, when the callback "onCreate()" is received, it causes the activity to be created. Then when the callback "onStart()" is received, that activity is moved from the Created state to the Started state. Likewise, the onPause() callback causes the activity to move from the Resumed state to the Paused state (this would happen when the user moves from the currently displayed app to another app).



Detailed view of the Activity Lifecycle and the system callbacks that move an activity from one state to another.

Process Management

- To keep the Android system running smoothly, app developers need to remain aware of the many ways in which an app can be terminated because the effect that doing so can have on resource allocation.
- For example, an app can be ended from any of three states: Paused, Stopped, or Destroyed. If the app is Paused after it was allocated an exclusive resource, such as a network connection or access to a critical device, these critical resources should be deallocated until it is time for the app to resume. To do otherwise is to risk the app closing prematurely (from the Paused state) still holding exclusive control of the resource. In other words, the app designer needs to make certain that each app ends gracefully whenever the user terminates it, no matter what state it happened to be in at the time.

Device Management

Design Factor	Sample Values	What It Means
Screen size	Phone: 3 in (5.1 cm) Tablet: 10 in (25 cm)	The screen's actual size as measured from one corner diagonally to the other.
Screen density	Phone: 164 dpi Tablet: 284 dpi	The number of pixels located within a certain area on the screen, such as a one inch line (dpi = dots per inch).
Orientation	Portrait (vertical) Landscape (horizontal)	The screen's orientation as viewed by the user.
Resolution	Phone: 240×320 Tablet: 1280×800	The number of physical pixels.

Four device display variables that directly impact the design of user interface screens.

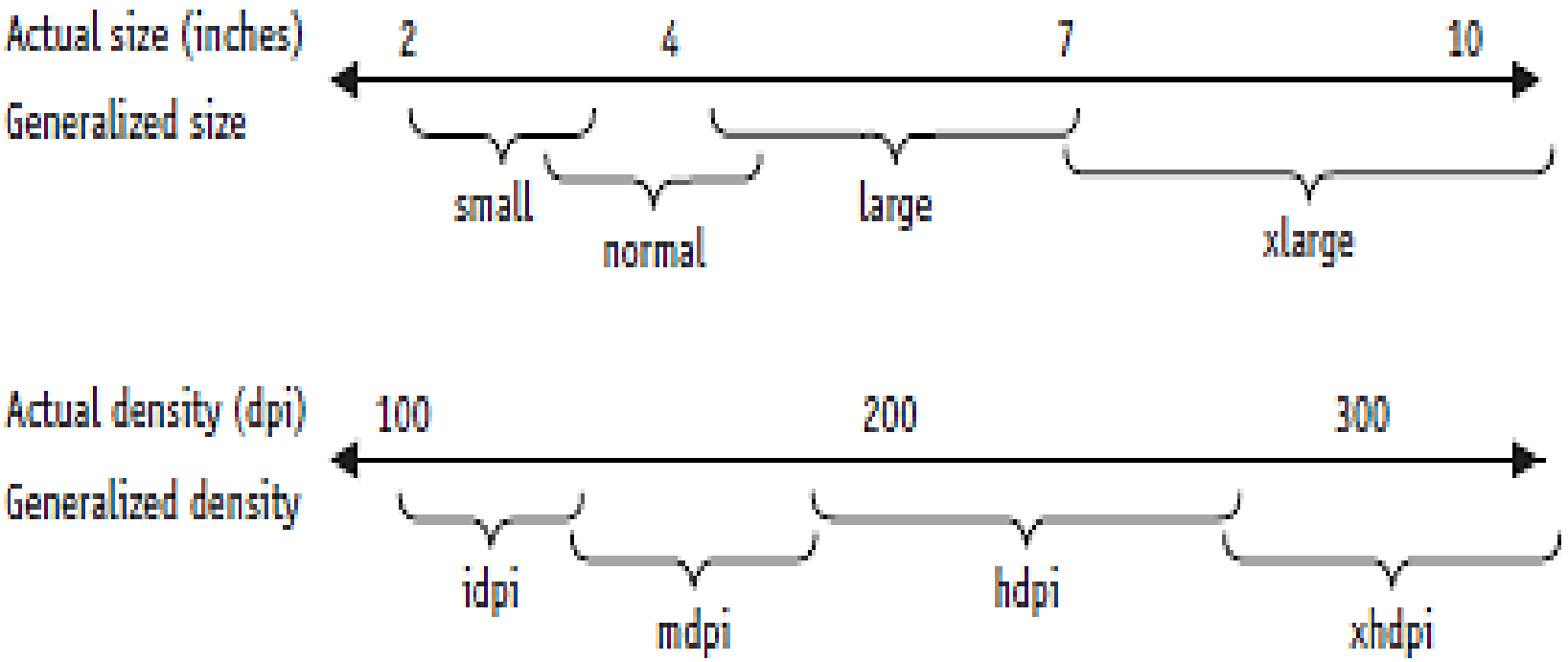
Device Management

- To aid designers, Android has introduced a fifth factor, called a density-independent pixel (dp), which is the equivalent to one physical pixel on a 160 dpi screen.
- App designers are encouraged to create interfaces using the dp unit. In this way, device and app designers can allow the system to perform the necessary scaling for each screen based on its size, resolution, orientation, and density.
- For example, if a designer is creating a simple game to run on all Android phones, the game could end up running on screens ranging from 240x320 dpi to 1920x1030 dpi in portrait mode (and (320x240 to 103x1920 in landscape mode). If the designer wrote the app using dpi units, the system would require coding for every possible screen combination. By writing the app using dp units, the same code can be used to display on every screen size.

Device Management

- Note that using dp units does not absolve the designer from stating in the app's manifest file the types of screens that are supported.
- Given the wide variation in screen configurations, designers may choose to create four different user interfaces to accommodate Android's four categories of screen sizes:
  - Extra-large screens: at least 960dp x 720dp
  - Large screens: at least 640dp x 480dp
  - Normal screens: at least 470dp x 320dp
  - Small screens: at least 426dp x 320dp
- The figure on the next page illustrates Android's four size categories and how their representative densities compare.

Device Management



Comparison of the four screen sizes supported by Android and the different densities for each size: low dpi (ldpi), medium dpi (mdpi), high dpi (hdpi), and extra high dpi (xhdpi).

Device Management

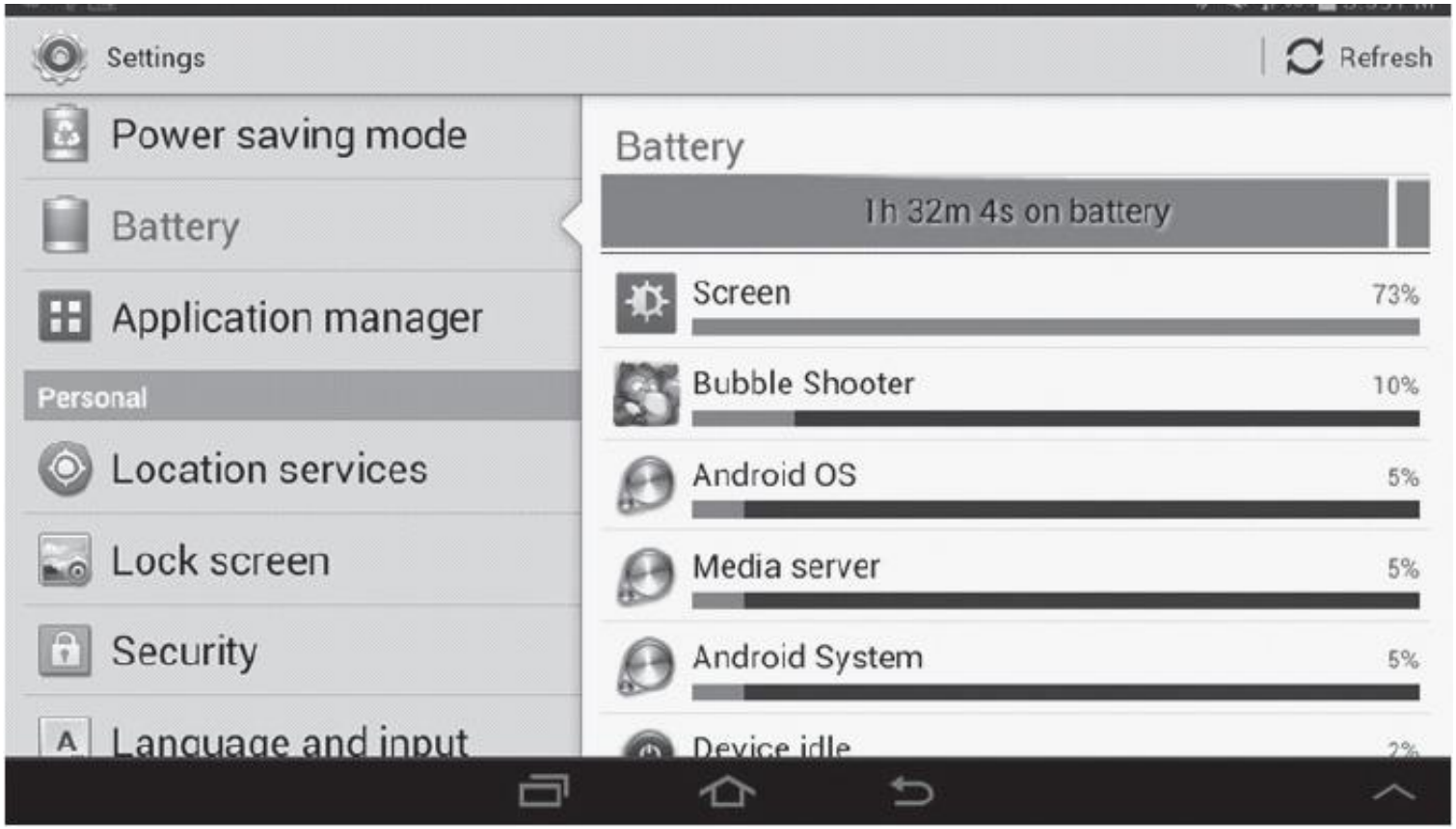
- The ultimate goal of app designers in Android is to give every user the impression that the app was designed specifically for the user's device, and not merely stretched or shrunk to accommodate various screens.
- Screen requirements and how to develop apps for them is a subject that is sure to change frequently.
- Current details about the Android user interface screen support can be found at:

  http://developer.android.com/guide/practices/index.html

Battery Management

- One of the key considerations for any OS running on a mobile device is management of the power supply, especially the battery.
- Battery usage information for an Android device is available from the Settings tab, as shown on the next page.
- To improve battery availability, users may choose to leave certain functions turned off until they actually need them, such as GPS, Bluetooth communications, background file syncing, live wallpaper, and haptic (vibration) feedback.
- In addition, using Wi-Fi instead of telephony can save power.
- Battery management is a field of its own and doesn't fall into the realm of OS. However, it is an ever-changing field which will have big impacts on future mobile devices.

Battery Management



This device has 1 hour, 32 minutes of battery time remaining. This display also indicates that the screen is consuming 73 percent of the device's battery resources at a very bright setting and that the Android operating system is using 5 percent.

File Management

- Routine file control in Android OS is managed by Linux at the kernel level. Therefore, each app has its own User ID, the part of the OS that is the user's own protected mode and that allows it to manage files it creates and executes. Therefore, if the app's developer does not explicitly expose it to other apps, no other apps are allowed to read or alter its files.
- However, if two apps are signed with the same digital certificate, then the two get the same User ID. As a result, the apps are allowed to share data, but this practice also means that the developer must take special precautions to make sure that they work correctly with each other.

File Management

- Users many need supplementary apps to perform the hand-on file manipulation that one might expect from a more robust OS.
- Currently, most “out of the box” Android devices gain the ability to move, copy, and otherwise manipulate files on the device and flash card, only by installing supplementary apps.

Security Management

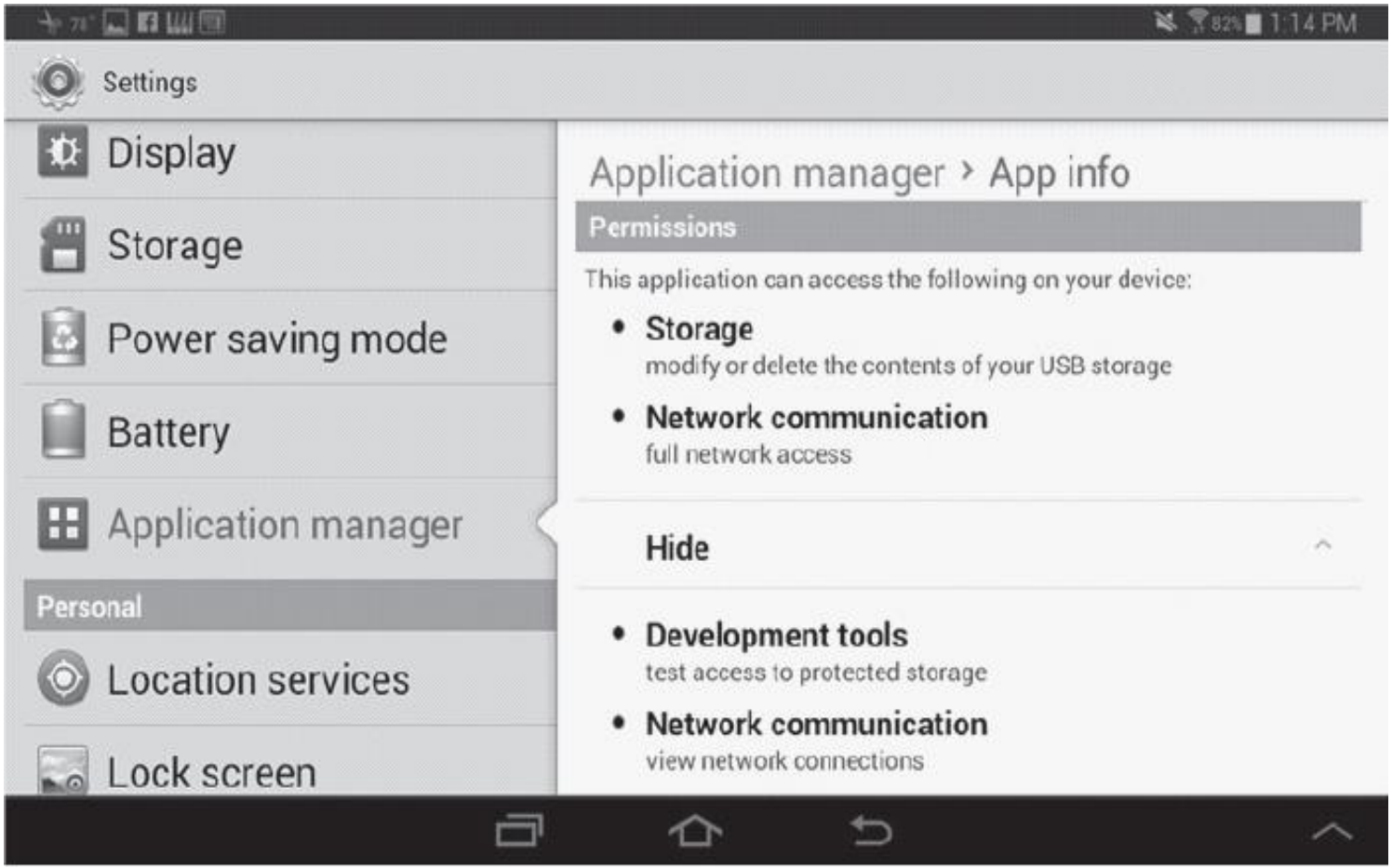
- The Android OS has a multiple tiered security structure designed to protect the user's data, protect the system's resources (including networking resources), and provide application isolation to prevent intentional damage or inadvertent vulnerabilities from a malicious or poorly designed app.
- There are two primary classes of Android apps: those that are pre-installed (such as calendar, camera, email, contacts, browser, and so on) and those installed by the user.
- The greatest vulnerabilities stem from those that are user installed.

Security Management

Permissions

- One critical aspect of Android security is user-defined permissions. Because the person who allows installation of an app is explicitly asked to grant permission for this app to access certain resources on the device, it is the individual who holds the key to device security.
- Before installing an app, the list of permissions is available for review, such as the display as shown on the next page. The app in question in the figure has requested access to only four permissions.

Security Management



This application requested these four permissions. The app is not installed until the user agrees to allow access.

Security Management

Permissions

- User granted permissions, a few of which are listed on the next page, are listed in the app's manifest.
- A survey done in 2012 found that the majority (72%) of the 400,000 apps that were examined, requested permissions that appeared to be outside of their apparent operational requirements.
- When that number is combined with the ability of the individuals to connect their personal devices to their organization's network, this can produce a wide range of vulnerabilities.

Security Management

Requested Permission	Application's Reason for the Request
Your location	Learn approximate user location using network information and/or precise GPS location.
Your accounts	Find accounts that have been set up for this device.
Personal Information	Read and/or modify contact lists, read and/or modify the call log, write to the dictionary.
Phone Calls	Read the status of the phone and identify callers.
Services that cost you money	Directly call phone numbers and send messages.
Network Communication	View and control network connections and/or Near Field Communication.

Before installing an app, the user is presented with a list of all the permissions requested by that application. A few are listed here.

Security Management

Device Access Security

- Android offers several levels of access protection so that users can choose how much protection they want for each mobile device, and these options can range from highest security to none, as shown in the table on the next page.
- Strong passwords are the highest built-in level of security, though even stronger security can be added through a high-security app.
  - The assumption is that the passwords chosen are not trivial and are not written down in plain sight.
  - Google recommends a two-step process to build a strong password consisting of numbers, letters, and symbols.
    - Start with a random phrase that is easy to remember.
    - Then insert random numbers, capital letters, and special characters to make it harder to guess.
  - More can be found at: http://www.google.com/safetycenter/

Security Management

Technique	Security Level
Password Matching	Highest security
PIN Matching	Medium to high security
Pattern Recognition	Medium security
Face Recognition	Low security
Finger Swipe	No security, open access

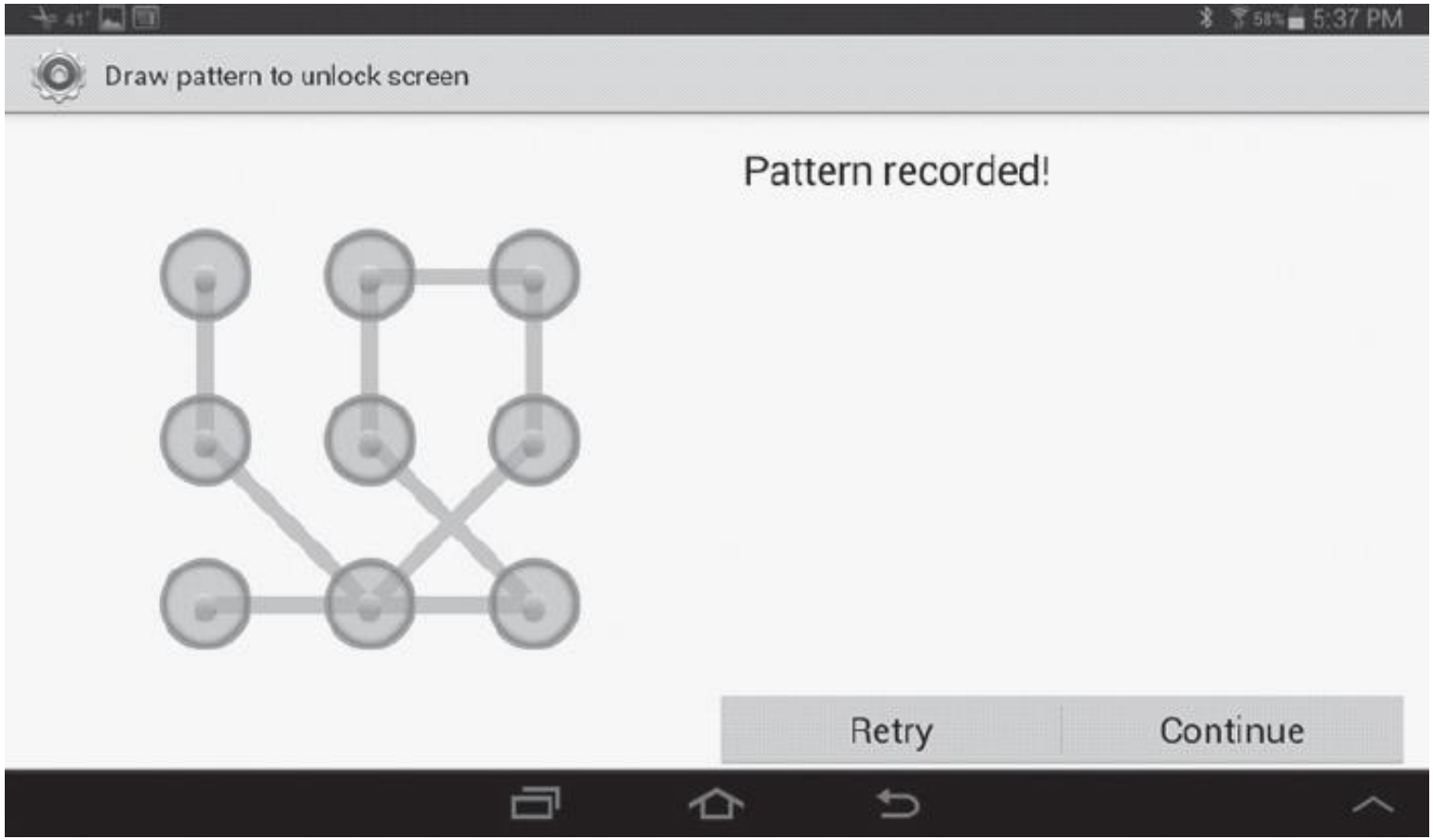
Each mobile device can be set to an appropriate level of the security.

Security Management

Device Access Security

- An alternative to password protection is Android's pattern recognition tool, which is similar to a graphical password.
- To set one up, the user chooses Settings, opens the Lock Screen security option, chooses Pattern to reach the screen shown on the next page, and follows the instructions to connect the nine dots in a certain pattern – one that can easily be remembered.
- Should an intruder attempt to use brute force to guess the pattern, the default setting allows only a limited number of incorrect tries before licking the device and requiring an alternative mode of access.

Security Management



To establish the pattern lock security option, the device owner traces a single line to connect the dots, such as the one shown here. Later, to unlock the device, the same pattern must be retraced exactly.

Security Management

Device Access Security

- Facial recognition is not currently considered a strong access control tool on Android because of several vulnerabilities.
- Expect future versions of Android to improve this area and it may ultimately become a strong access control mechanism.

Encryption Options

- Android offers encrypted storage options, as shown in the figure on the next page. Once an account, setting, application, data, picture or other file is encrypted, the user will be required to enter a valid password with every sign-on.
- This option is also available from the Settings/Security menu.

Security Management

For higher security, encryption is offered. It's a time-consuming option to invoke, so the device owner should have the device plugged in or at least fully charged before starting it.