

UNIT I

INTRODUCTION

Characteristics and advantages of mobile communication,
types of mobile applications –development approaches, overview of mobile
strategy and designing mobile solutions.

Mobile computing project structure, building and testing.

Evolution of Modern Mobile Wireless communication system.

Mobile computing vs. wireless - MAC Protocols –Wireless
MAC Issues – Fixed Assignment Schemes – Random Assignment Schemes –
Reservation Based Schemes.

What is Mobile Computing?

- What is computing?

The capability to automatically carry out certain processing related to service invocations on a remote computer .

- What is the mobility ?

The capability to change location while communicating to invoke computing service at some remote computers.

- What is mobile computing?

Ability to compute remotely while on the move. It is possible to access information from anywhere and at anytime.

- ▶ A simple definition could be:
Mobile Computing is using a computer (of one kind or another) while on the move
- ▶ Another definition could be:
Mobile Computing is when a (work) process is moved from a normal fixed position to a more dynamic position.
- ▶ A third definition could be:
Mobile Computing is when a work process is carried out somewhere where it was not previously possible.
- ▶ Mobile Computing is an umbrella term used to describe technologies that enable people to access network services anyplace, anytime, and anywhere.

Advantages of mobile communication

Emergencies

A mobile communication device can be helpful in case of an emergency

Sharing Information

With hand-held communications devices, business professionals can instantly share information with clients

Disadvantages

- Safety Concerns-use of mobile device can be dangerous due to distractions
- Less Down Time-Because many business professionals are connected to clients through cellular devices, there is no down time anymore.

Characteristics of Mobile Computing

- **Ubiquity** – The ability of a user to perform computation from anywhere and at anytime.
- **Location awareness** – Many applications requires or value additions by location based services.
- **Adaptation**- Ability to adjust to bandwidth fluctuations without inconveniencing the user.

- **Broadcast**- Efficient delivery of data can be made simultaneously to hundreds of mobile users
- **Personalization** – Services in a mobile environment can be easily personalized according to a user's profile.

Mobile computing vs Wireless networking

- ❖ Mobile computing is based on wireless networking and helps to invoke computing services on remote servers while on the move.
- ❖ So Wireless networking is an important and necessary ingredient of mobile computing.
- ❖ Mobile computing also requires the applications themselves – their design and development, the hardware at the client and server sides

Types of mobile applications

- ❑ Native Apps
- ❑ Mobile Web Apps
- ❑ Hybrid Apps



Native Apps

- ❑these apps are on the device.
- ❑They are accessed through the icons on the home screen.
- ❑Such apps are installed through an application store like Google play and Apple's app store.
- ❑These apps can also use other apps like camera, music, GPS, contact list and location etc.
- ❑Eg:Pokemon Go, Twitter, and Waze,

Native app advantages

- Complete access to device hardware, APIs
Access to built-in features of the device
- Installable, can be app store deployed /Available from app stores
- Native UI
SDK for developers
- Powerful platform-specific development and debugging tools direct from platform vendors

Native app disadvantages

- Multiple implementations required to reach multiple platforms
- Multiple skill sets and programming languages
- Requires installation
- New tools needed to manage app security, enforce data security policies
- High price and long development time

Mobile Web Apps

- Mobile web apps are not actually applications but they are websites which give an illusion of native apps.
- They run on the browsers and are written in HTML5.
- they are built with the consideration to fit into different screen sizes of mobile devices.

Web app advantages

Build the development team fast

Support every device, every platform and every version of OS

Fast deployment of new features

No need to support multiple versions of the software

No app store approval

Visible to search engines

Web app disadvantages

- Limited access to device hardware, APIs(device features)
- Poor offline support, requires “always on” Internet connection
- Unable to “install” on a device or publish via an app store
- Ad blockers

Hybrid Apps

- As the name suggests are a mix of both native and web apps.
- Like the native app, they are found on the app stores and they can also take advantage from other mobile features available.
 - They also rely on the HTML in the browser.
- Often **App Design Company built a hybrid app** as it is a profitable affair because then you don't have to make a separate app.
- Amazon (for iOS and Android), Evernote (for iOS and Android), Netflix (for iOS and Android)

Hybrid app ctd..

Hybrid development is best used when the requirements of an app exceed the limits of web, but do not demand the full power of native."

Hybrid app advantages

Cross-platform(One code base for all platforms)

Web development technology

Lower price

Reaching a wider audience

Downloadable from the app store & can be installed

No ad blockers

- Easy to transition from web to hybrid development, reuse code
- Extensive access to device hardware, APIs

Hybrid app disadvantages

Performance limited by web's capabilities

- Requires installation
- Not native look'n'feel

Native Mobile App

- IOS - Developed using Objective-c
- Android - Developed using JAVA
- Need to Install from APP Store.
- Available as an Application on Device.

Mobile Web App

- Developed using typical web development technology - HTML, CSS, Java Script.
- View size of the Web page fit to the real-estate of the device.
- Accessed through the browsers on the device

Hybrid Mobile App

- Wrapping the HTML and creating Native like look and feel (HTML within the app itself). Framework like Phone Gap support this development.
- Native Mobile App with Web view control and render the HTML directly on the web view (HTML Rendered from enterprise server).
- View size of the Web page fit to the real-estate of the device.
- Accessed through the browsers on the device

Mobile Application Development Approaches:

- ✓ Native Application Development Process
- ✓ Web Application Development Process
- ✓ Hybrid Application Development Process

Native Application Development Process

- Native applications are built using vendor specific programming languages and development Toolkits
- They are binary executable files that are installed through an app store.
- Developers write the source codes and compile it to binary forms.
- They have full access to the hardware functionalities of the device.

Essential skills*: Objective-C, Java, .NET

Essential tools*: XCode (for iOS), Eclipse (for Android), Visual Studio (for WinPhone)

Platform reach: Each app only reaches one platform

Sharable cross-platform codebase:
0% (No UI, No logic)



Android Development
iOS Development
Windows Development

Web Application Development Process

Essential skills: HTML, JavaScript, CSS

Essential tools: Anything capable of developing web apps

Platform reach: iOS, Android, Windows Phone or any HTML5 capable mobile browser

Sharable cross-platform codebase:
100% (UI + Logic)

Hybrid app development process

- Essential skills: HTML, JavaScript, CSS,
- Hybrid container (such as Apache Cordova)
- Essential tools: Anything used for web development* + hybrid SDKs
- Platform reach: Limited to reach of hybrid container, but most reach all major platforms
- Sharable cross-platform codebase:
 - Almost 100% (Some platform specific UI may be desired)

HOW DO YOU CHOOSE THE “RIGHT” APPROACH?

- Who is the audience for the app?
- How long do we have to develop the app and for how many platforms?
- What are the skills of our development team?
- Does the app need to work offline?
- Does the app need to access device APIs or hardware features?
- What is most important for the app: Experience, Reach or Cost?

An overview of mobile strategy

- What is a mobile strategy??
- More screens= profit.
- This used to be how companies approached mobile apps.

An overview of mobile strategy

1. Define Your Objectives:

- to be specific about your goals
- setting a **firm deadline**.

2. Select the Right Mobile Environment web, hybrid or native application.

- two options: a mobile site or a mobile application.
- Mobile site- *appear inside the browser on any internet-enabled mobile device. For example, the Apple iPhone uses the Safari browser.*
- Mobile applications -*require a device-specific download from a marketplace, such as the Apple App Store or the Android Market.*

3 Design for User Experience

- ease of use
- design
- performance
- Functionality

4 Importance of Development Quality

5. Integrate With Social Media

- sharing reviews,
- having social sharing buttons
- promoted through social media ads

6. Take Advantage of the Multi-Screen/Multi-Device Opportunity

Need to do marketing campaigning by device,location etc

6. Send Out Timely Alerts and Notifications

7. Use Location Based Ads

8. Optimize Your Emails

Structure of Mobile Computing Application

- A mobile computing application is usually structured In terms of the functionalities implemented .
- As shown in the figures 2.3 and 2.4 the three tiers are named presentation tier, application tier and data tier.

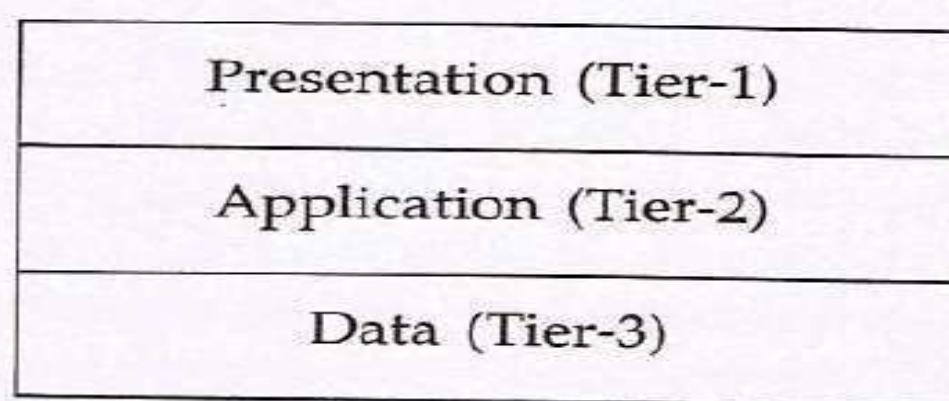


Figure 2.3 *The three tier structure of a mobile computing application.*

**Presentation
Tier**

Find the
sales
total

Five total
sales

Tier 1

**Application
Tier**

Find the list of sales
made in last year

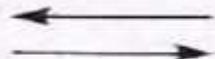
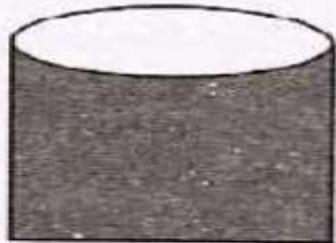
Query

Sum of all the sales

Sales 1
Sales 2
Sales 3
Sales 4
Sales 5

Tier 2

**Data
Tier**



Storage

Tier 3

Figure 2.4 *Functionalities provided by each tier structure of a mobile computing application.*

- **Presentation tier** – The topmost level, concerns the user interface. It facilitates the users to issue requests and to present the results meaningfully. It includes web browsers and client programs.
- **Application tier** – It is responsible for making logical decisions and performing calculations. Implemented using Java, .NET services, cold fusion etc. Implemented on a fixed server.
- **Data tier** – Responsible for data storage, access and manipulation. Implemented on a fixed server.

Cellular mobile communication

- Area of coverage split into cells
- Cell shapes are irregular
- Cells do overlap to some extent
- Base station(BS) located at the centre of each cell
- BS receives communication from all mobile handsets in the cell
- BS forwarded data to appropriate handset.
- Hands-off: when mobile handsets moves from one cell to another ,BS hands-off the call to the BS in the new cell

Generation of cellular technologies

- ✓ 1G
- ✓ 2G/2.5G
- ✓ 3G
- ✓ 4G
- ✓ 5G

1G TECHNOLOGY

- 1G refers to the first generation of wireless telephone technology, mobile telecommunications which was first introduced in 1980s and completed in early 1990s
- Its Speed was upto 2.4kbps.
- It allows the voice calls in 1 country.
- 1G network use Analog Signal
- AMPS was first launched in USA in 1G mobile systems.



DRAWBACKS OF 1G

Poor Voice Quality Poor Battery Life

Large Phone Size No Security

Limited Capacity

Poor Handoff Reliability

1G Wireless System



2G TECHNOLOGY

- *2G technology refers to the 2nd generation which is based on GSM.*
- *It was launched in Finland in the year 1991.*
- *2G network use digital signals.*
- *It's data speed was upto 64kbps.*

Features Includes:

- | *It enables services such as text messages, picture messages and MMS (multi media message).*
- | *It provides better quality and capacity .*





DRAWBACKS OF 2G

- *2G requires strong digital signals to help mobile phones work.*
- *If there is no network coverage in any specific area , digital signals would weak.*
- *These systems are unable to handle complex data such as Videos*



2G Wireless System

2.5G TECHNOLOGY

- ❑ *2.5G is a technology between the second (2G) and third (3G) generation of mobile telephony.*
- ❑ *2.5G is sometimes described as 2G Cellular Technology combined with GPRS.*
- ❑ **Features Includes:**
 - ❑ *Phone Calls*
 - ❑ *Send/Receive E-mail Messages*
 - ❑ *Web Browsing*
 - ❑ *Speed : 64-144 kbps*
 - ❑ *Camera Phones*
 - ❑ *Take a time of 6-9 mins. to download a 3 mins. Mp3 song*



3G TECHNOLOGY

3G technology refer to third generation which was introduced in year 2000s.

- *Data Transmission speed increased from 144kbps- 2Mbps.*
- *Typically called Smart Phones and features increased its bandwidth and data transfer rates to accommodate web-based applications and audio and video files.*



FEATURES OF 3G TECHNOLOGY

Providing Faster Communication

Send/Receive Large Email Messages

High Speed Web / More Security Video Conferencing /

3D Gaming

TV Streaming/ Mobile TV/ Phone Calls

Large Capacities and Broadband Capabilities

11 sec - 1.5 min. time to download a 3 min Mp3 song.



DRAWBACKS OF 3G TECHNOLOGY

- ❑ *Expensive fees for 3G Licenses Services*
- ❑ *It was challenge to build the infrastructure for 3G*
- ❑ *High Bandwidth Requirement*
- ❑ *Expensive 3G Phones.*
- ❑ *Large Cell Phones*



4G TECHNOLOGY (Anytime ,Anywhere)

- ❑ *started from late 2000s.*
- ❑ *Capable of providing 100Mbps – 1Gbps speed.*
- ❑ *One of the basic term used to describe 4G is MAGIC.*
- ❑ *MAGIC:*
 - ❑ *Mobile Multimedia*
 - ❑ *Anytime Anywhere*
 - ❑ *Global Mobility Support*
 - ❑ *Integrated Wireless Solution*
 - ❑ *Customized Personal Services Also known as Mobile Broadband Everywhere.*

4G (Anytime, Anywhere)

- *The next generations of wireless technology that promises higher data rates and expanded multimedia services.*
- *Capable to provide speed 100Mbps-1Gbps. High QOS and High Security*
- *Provide any kind of service at any time as per user requirements, anywhere.*
- **Features Include:**
 - *More Security*
 - *High Speed*
 - *High Capacity*
 - *Low Cost Per-bit etc.*



DRAWBACKS OF 4G

- ❑ *Battery uses is more*
- ❑ *Hard to implement*
- ❑ *Need complicated hardware Expensive equipment*

required to implement next generation network.



5G TECHNOLOGY

- *which was started from late 2010s have not been deployed.*
- *Complete wireless communication with almost no limitations.*
- *It is highly supportable to WWW (Wireless World Wide Web).*



BENEFITS OF 5G TECHNOLOGY

- *High Speed, High Capacity*
- *5G technology providing large broadcasting of data in Gbps .*
- *Multi - Media Newspapers, watch T.V programs with the clarity as to that of an HD Quality.*
- *Faster data transmission than of the previous generations.*
- *Large Phone Memory, Dialing Speed, clarity in Audio/Video.*
- *Support interactive multimedia , voice, streaming video, Internet and other*
- *5G is More Effective and More Attractive.*



MEDIA ACCESS CONTROL(MAC)

- A channel-access scheme is also based on a multiple access protocol and control mechanism, also known as media access control (MAC). This protocol deals with issues such as addressing, assigning multiplex channels to different users, and avoiding collisions.

MAC protocol properties

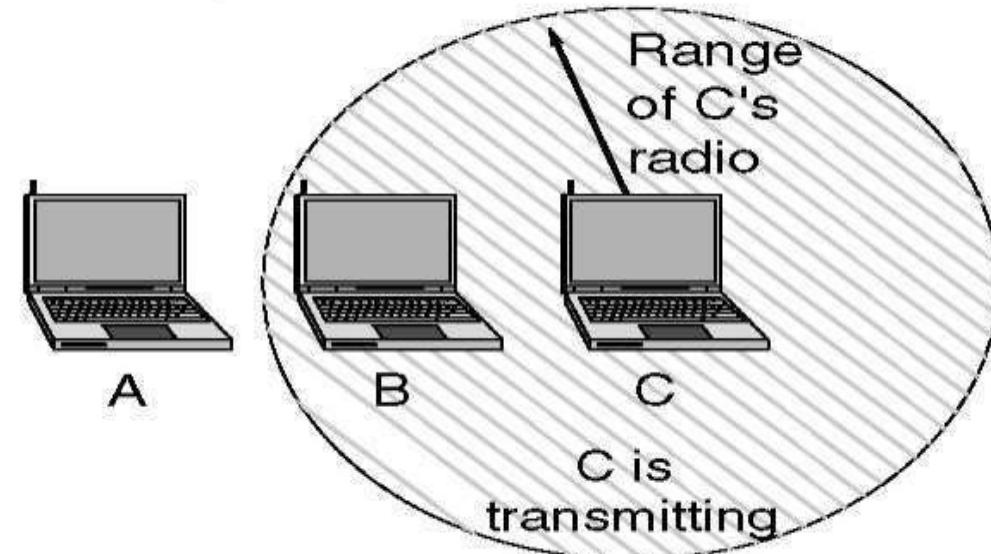
- Should implement rules to enforce discipline for a shared channel
- Maximize utilization of the channel
- Fair channel allocation
- Capable of supporting different types of traffic with different maximum & average bit rate
- Should be Robust in equipment failures & network condition changes

MAC protocol Issues

The Hidden Terminal Problem

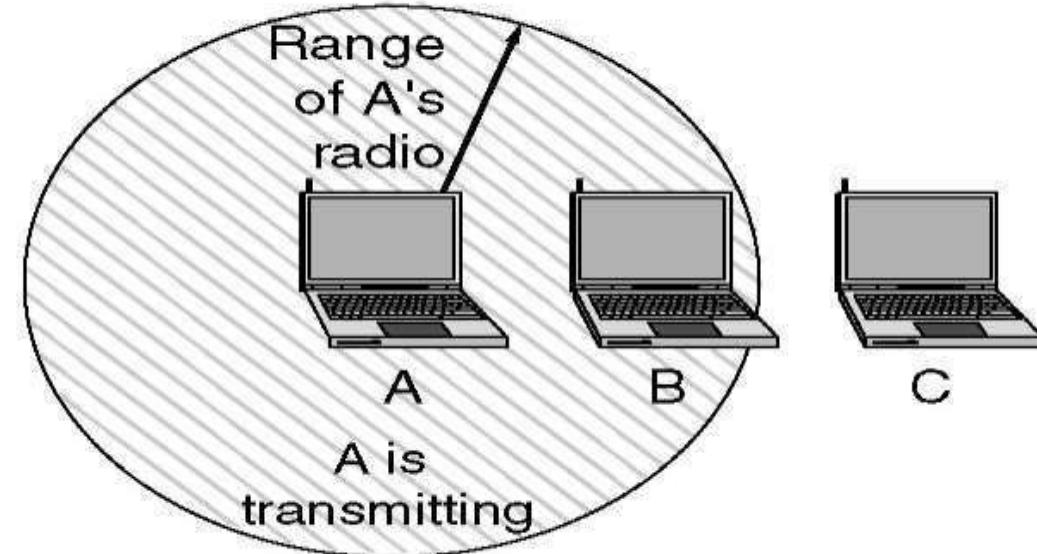
- Wireless stations have transmission ranges and not all stations are within radio range of each other.
- Simple CSMA will not work!
- C transmits to B.
- If A “*senses*” the channel, it will not hear C’s transmission and falsely conclude that A can begin a transmission to B.
- Create a very difficult and important arbitration problem that a MAC protocol needs to resolve.

A wants to send to B
but cannot hear that
B is busy



(a)

B wants to send to C
but mistakenly thinks
the transmission will fail



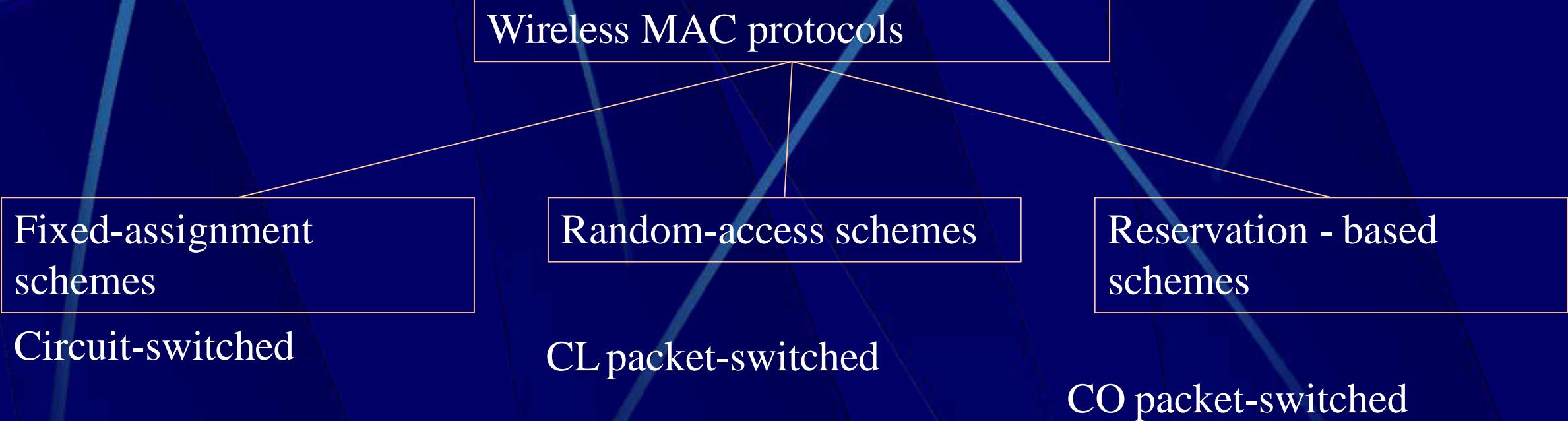
(b)

Figure 4-26.(a)The hidden station problem. (b) The exposed station problem.

The Exposed Station Problem

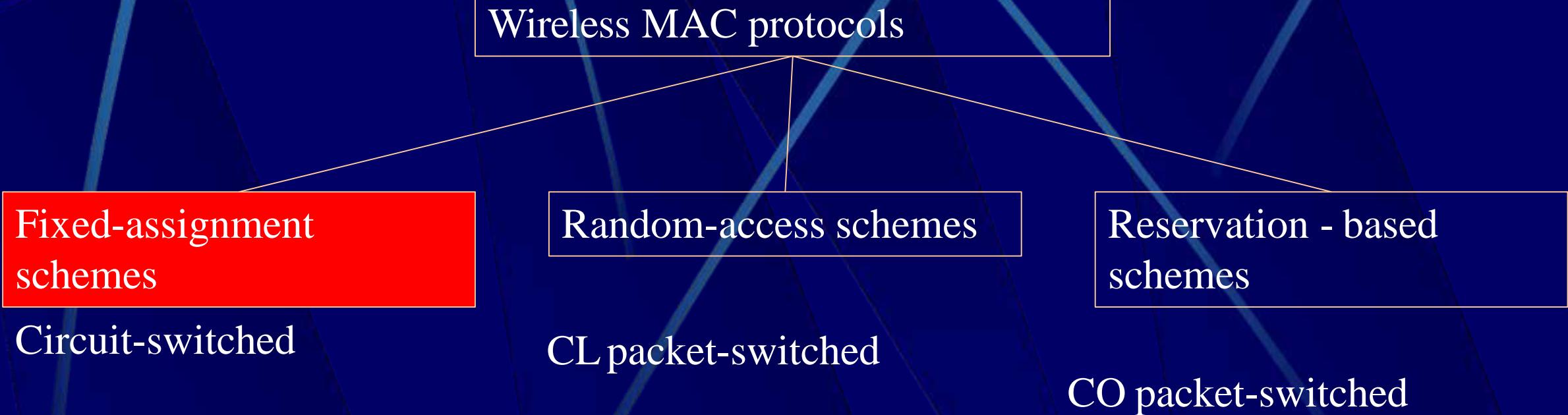
- ❖ The inverse problem.
- ❖ B wants to send to C and listens to the channel.
- ❖ When B hears A's transmission, B falsely assumes that it cannot send to C.
- ❖ It leads to inefficient spectrum usage as well as unnecessary transmission delays.

Classification of wireless MAC protocols



CL – Connection Less. CO – Connection Oriented

Classification of wireless MAC protocols

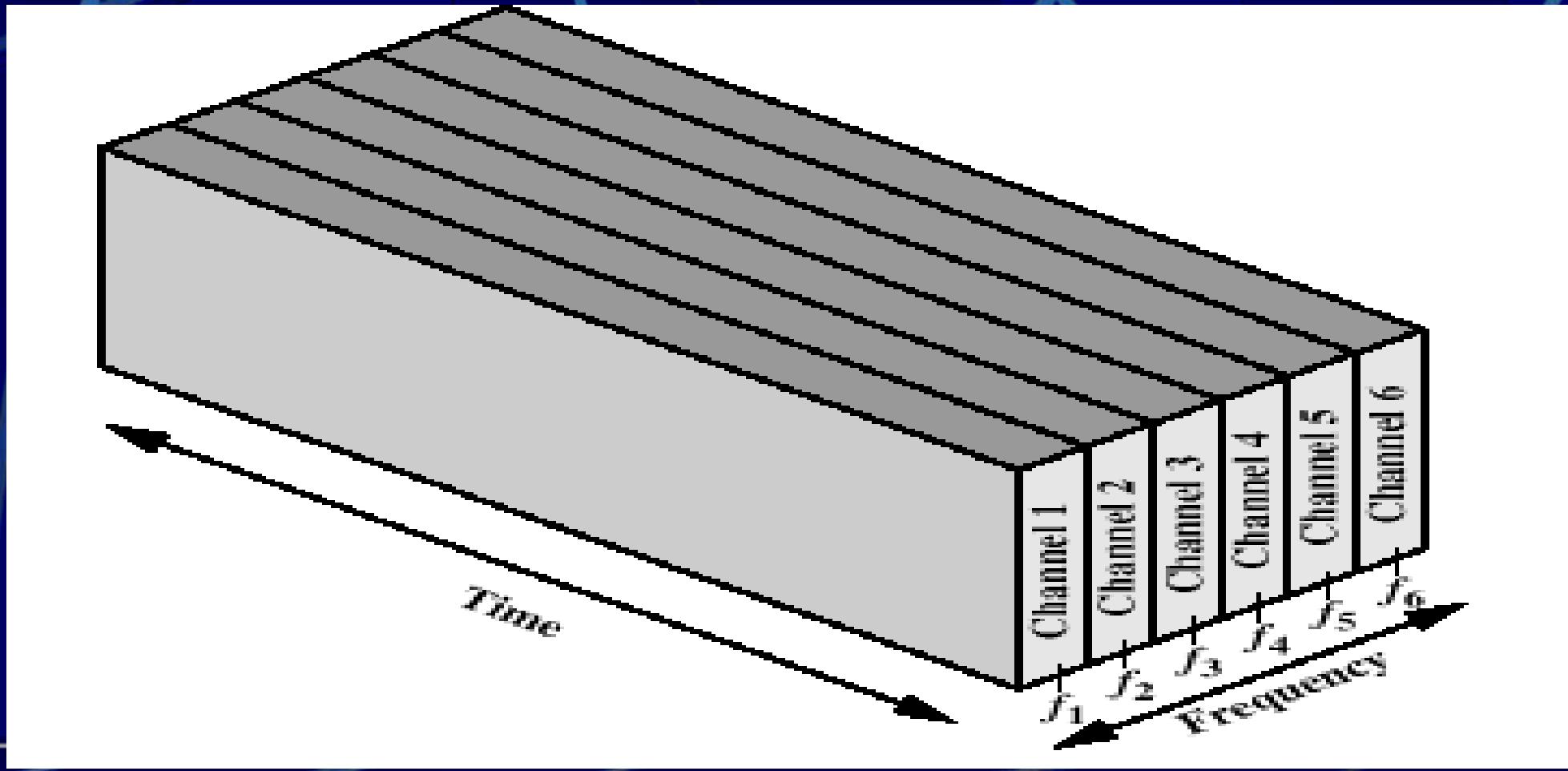


CL – Connection Less. CO – Connection Oriented

Frequency Division Multiple Access (FDMA)

- ❖ Frequency range divided into many narrower frequency bands called channels.
- ❖ In an FDMA system, each user has its own frequency channel.
- ❖ Most full duplex FDMA systems must transmit and receive simultaneously(MS-BS & BS-MS).
- ❖ It does not achieve a high channel utilization due to idle channel.

Frequency Division Multiple Access



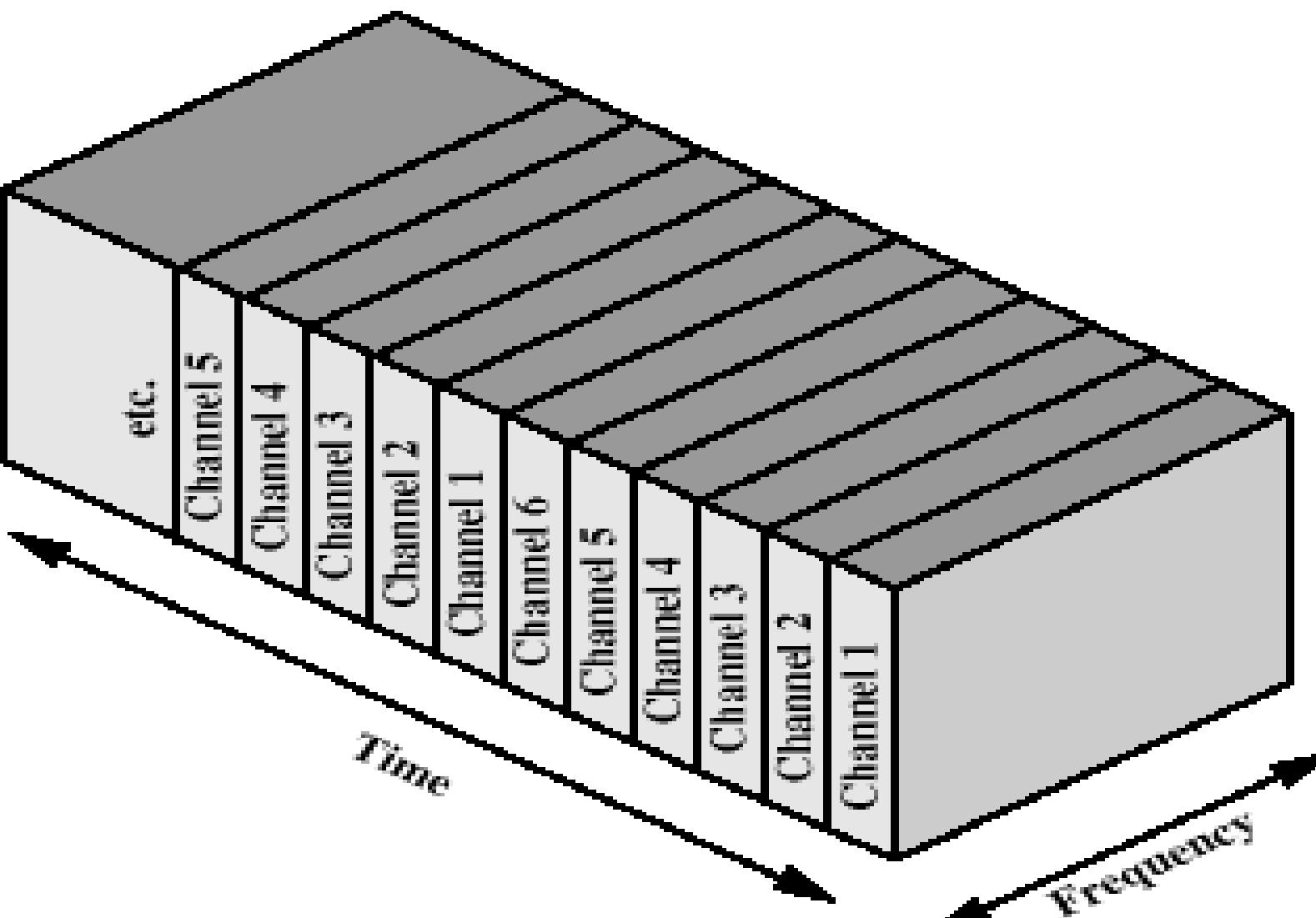
FDMA



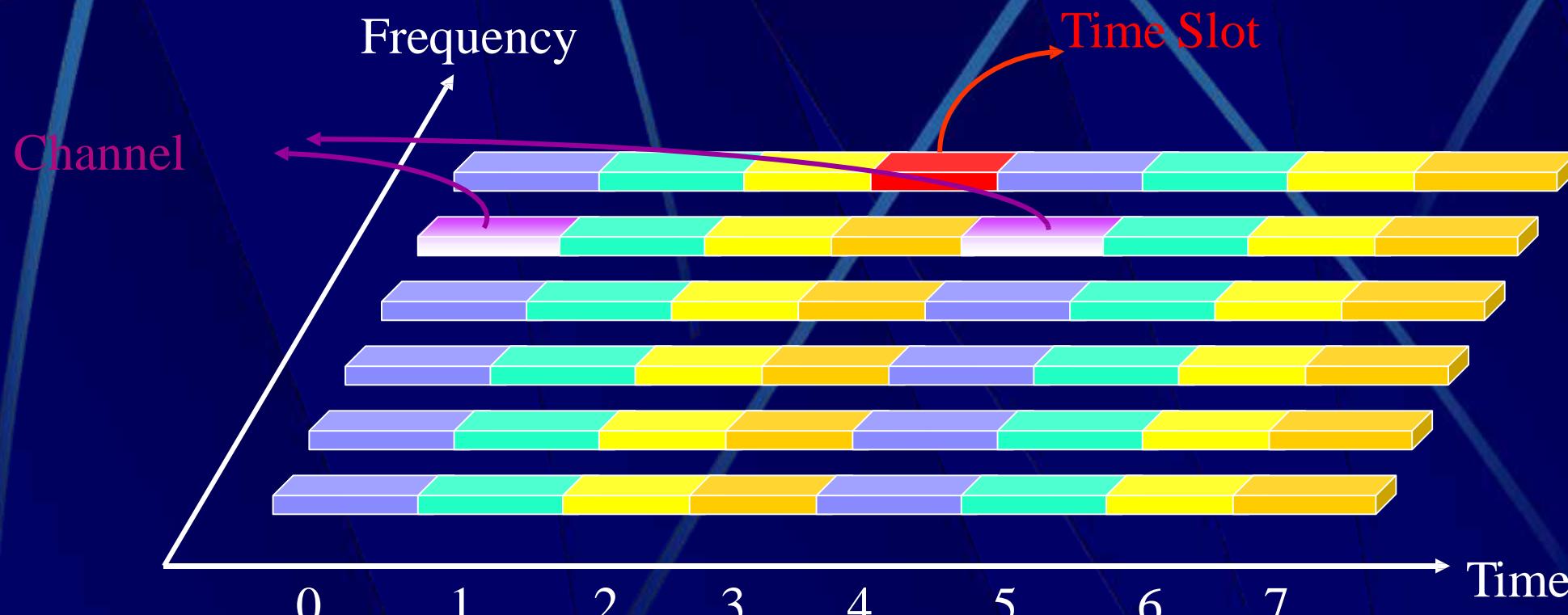
Time Division Multiple Access (TDMA)

- In TDMA, a set of N users share the same radio channel, but each user only uses the channel during predetermined slots.
- A frame consists of N slots, one for each user. Frames are repeated continuously.
- Time slots are allocated to users in a round robin manner .
- Unused time slots go idle, leading to low channel utilization

Time-division multiplexing



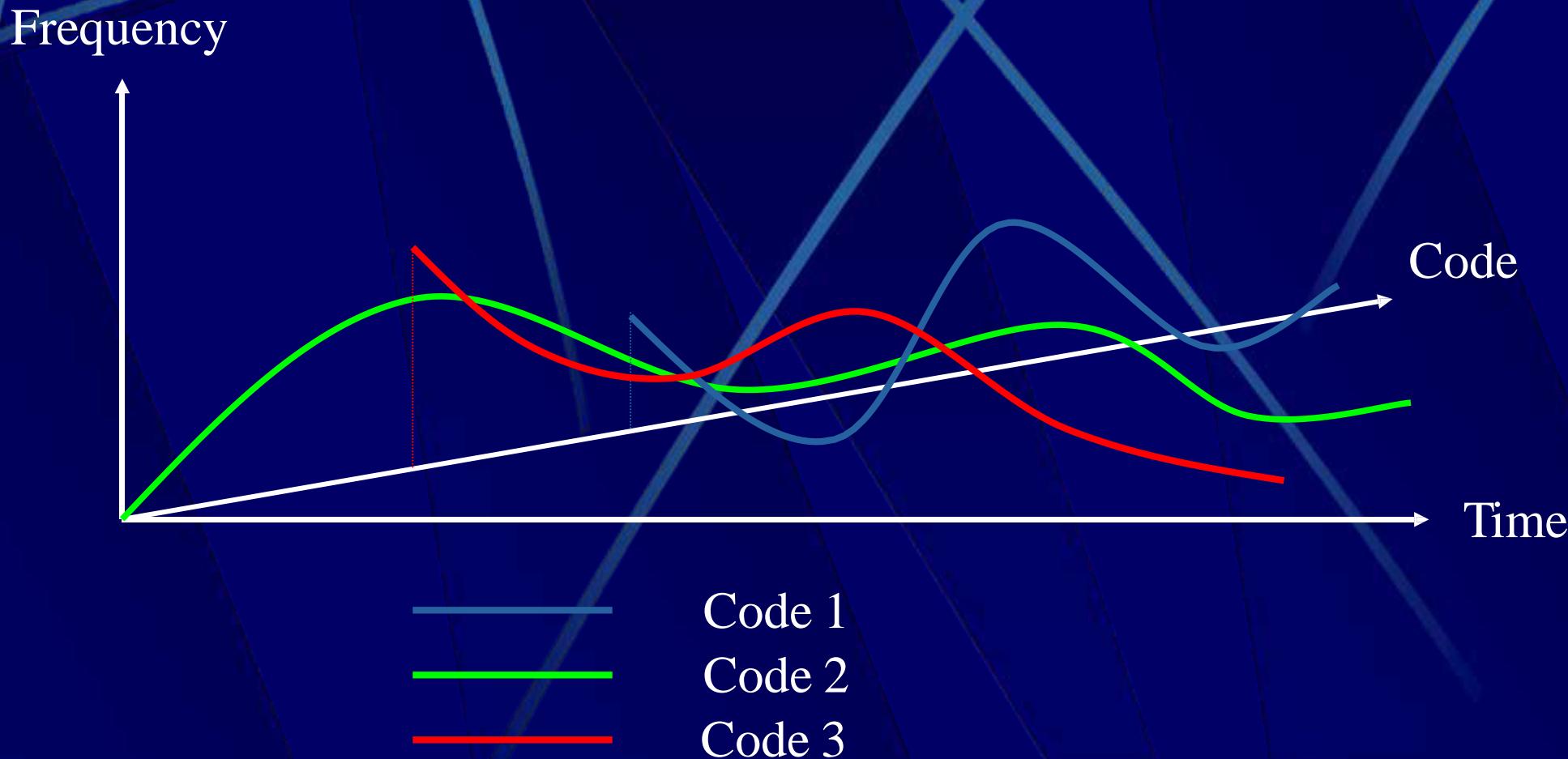
TDMA



Code Division Multiple Access (CDMA)

- Multiple users use the same frequency at the same time.
- Users are allotted different codes(0 & 1) to access same channel.
- All the senders send signals simultaneously.
- The signals can be distinguished from each other by frequency spreading code known as the m bit pseudo-noise(PN) code sequence.
- Using m bits $2^m - 1$ codes obtained
- Each user will use only one code.

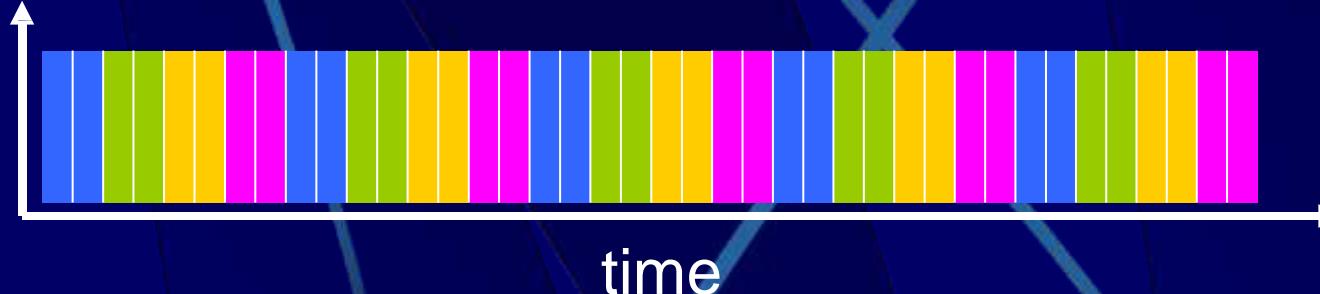
CDMA



Channel Partitioning MAC protocols

TDMA: Time Division Multiple Access

Frequency

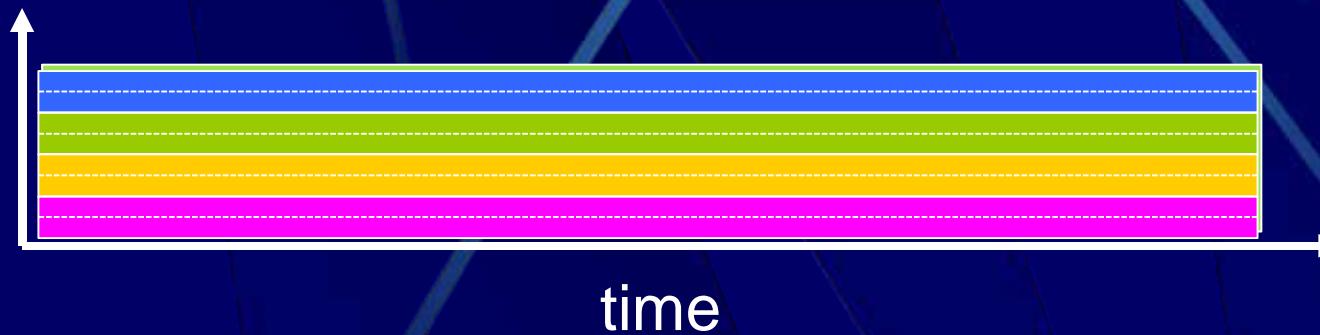


Example:
4 users



FDMA: Frequency Division Multiple Access

Frequency



CDMA: Code Division Multiple Access

- Same frequency and time but different codes.

Classification of wireless MAC protocols



CL – Connection Less. CO – Connection Oriented

Random access MAC Schemes

- Number of random assignment schemes . A few important.
- ALOHA
- Slotted ALOHA
- CSMA
- CSMA/CD
- CSMA/CA

ALOHA

- Simplest scheme
- When a node has data to send it begin to transmit
- Does not check whether channel is busy before transmitting.
- If the frame reached destination then next frame sent. Otherwise resend the frame.
- This scheme is used when less number of senders send data.
- Collision become high if transmission is high.

Slotted ALOHA

- ❖ An improvement over pure ALOHA..
- ❖ Chances of collision are reduced
- ❖ Time is divided into equal sized slots in which a packet can be sent. The size of packet is restricted.
- ❖ Send packet only at the beginning of a slot.
- ❖ Employ beacon signals to mark the beginning of a slot.
- ❖ Does not work well if the number of stations contending to send data is high.
- ❖ In such case CSMA scheme works better.

Carrier Sense Multiple Access CSMA

- ❖ Carrier Sense Multiple Access
 - ❖ sense carrier
 - ❖ if idle, send
 - ❖ wait for ack
 - ❖ If there isn't one, assume there was a collision, retransmit

Extension of CSMA

- ❖ The extension of CSMA are the collision detection CSMA/CD and the collision avoidance CSMA/CA techniques.
- ❖ Why CA and CD?
 - ❖ Difficult to detect collisions in a wireless network – why?
 - ❖ A transmitting station cannot effectively distinguish incoming weak signals from noise and the effects of its own transmission;
 - ❖ Hidden station problem:
 - ❖ Two mutually far away stations A and C want to send to B.
 - ❖ At A and C, channel appears idle
 - ❖ But collision occurs at B

CSMA/CD

- CSMA/CD multi-access control protocol.
 1. Each station listens before it transmits.
 2. If the channel is busy, it waits until the channel goes idle, and then it transmits.
 3. If the channel is idle it transmits immediately. Continue sensing.
 4. If collision is detected, transmit a brief jamming signal(NAK), then end transmission, wait for a random time, and retransmit.
 - collision detection is not by waiting for an ACK

CSMA/CA

- ❖ During the time a node is transmitting on the channel, several nodes might be wanting to transmit and waiting for it to become free.
- ❖ The moment the transmitting node completes its transmission and would all starts transmitting at the same time.
- ❖ To overcome in the collision avoidance scheme, all nodes are forced to wait for a random time and then sense the medium again before starting their transmission.
- ❖ If the medium is sensed to be busy, further random amount of time and so on.
- ❖ Thus the chance of two nodes starting to transmit at the same time would be greatly reduced.

Classification of wireless MAC protocols

Wireless MAC protocols

Fixed-assignment
schemes

Circuit-switched

Random-access schemes

CL packet-switched

Reservation - based
schemes

CO packet-switched

CL – Connection Less. CO – Connection Oriented

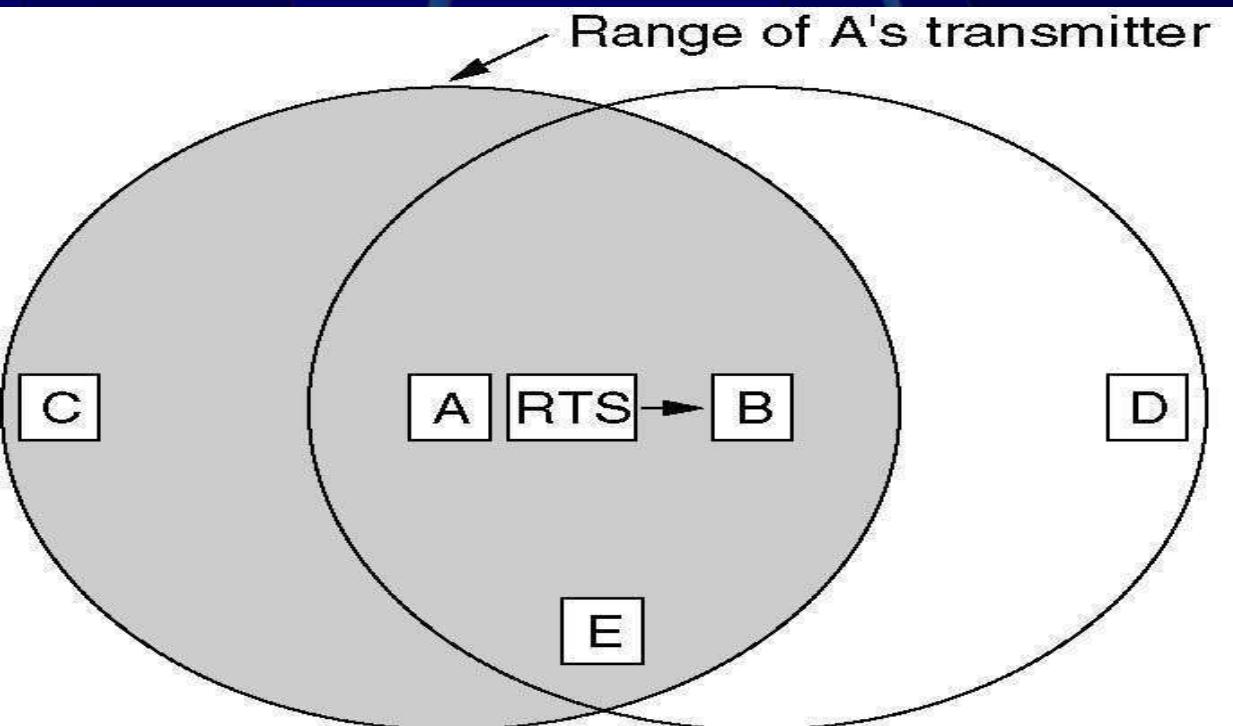
Reservation based schemes

- Basic form of the reservation scheme is RTS/CTS scheme.
- A sender transmits an RTS (Ready to Send) packet to the receiver before the actual data transmission.
- On receiving this the receiver sends CTS (Clear to Send) packet.
- The actual data transfer commences only after that.
- The other nodes sharing the medium sense the CTS packet, they stop from transmitting until the transmission from the sending node is completes.

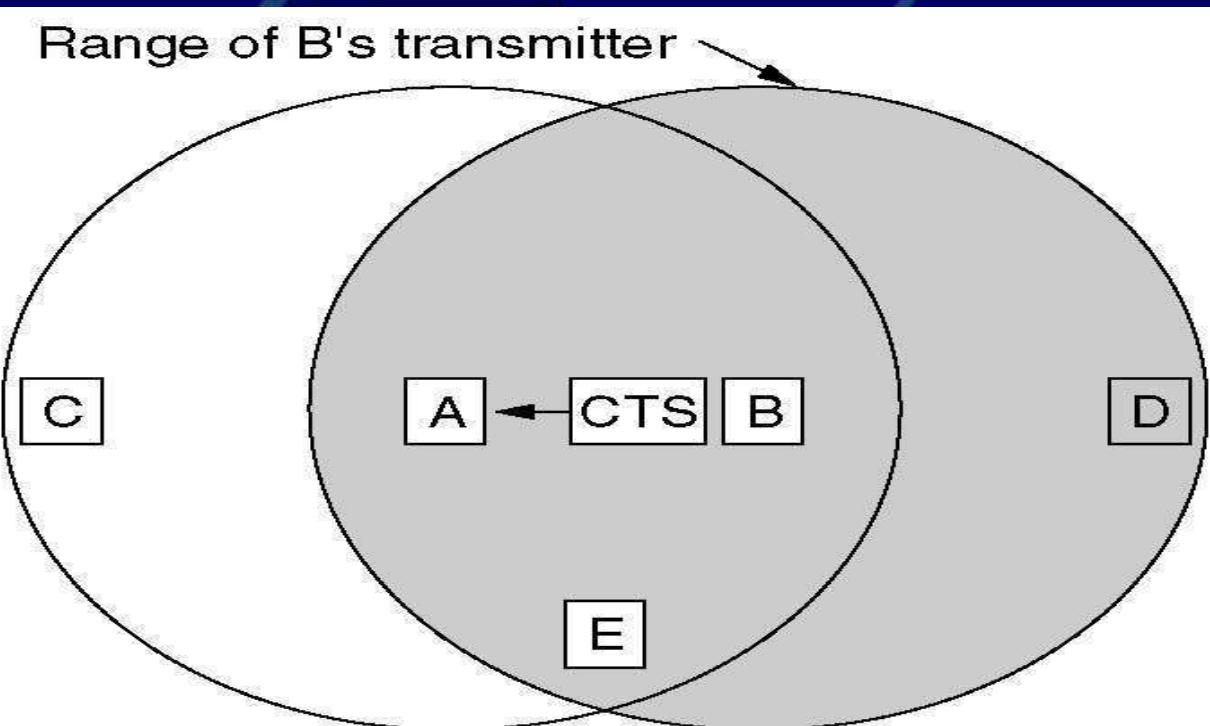
Contention-based protocol

- MACA(Multiple Access Collision Avoidance) Protocol
- MACA solves the hidden/ exposed terminal problems
 - When a node wants to transmit a data packet, it first transmit a **RTS (Request To Send)** frame.
 - The receiver node, on receiving the RTS packet, if it is ready to receive the data packet, transmits a **CTS (Clear to Send)** packet.
 - Once the sender receives the CTS packet without any error, it starts transmitting the data packet.
 - If a packet transmitted by a node is lost, the node uses the binary exponential back-off (BEB) algorithm to back off a random interval of time before retrying.
- The binary exponential back-off mechanism used in MACA might starves flows sometimes. The problem is solved by MACAW.

MACA Protocol



(a)



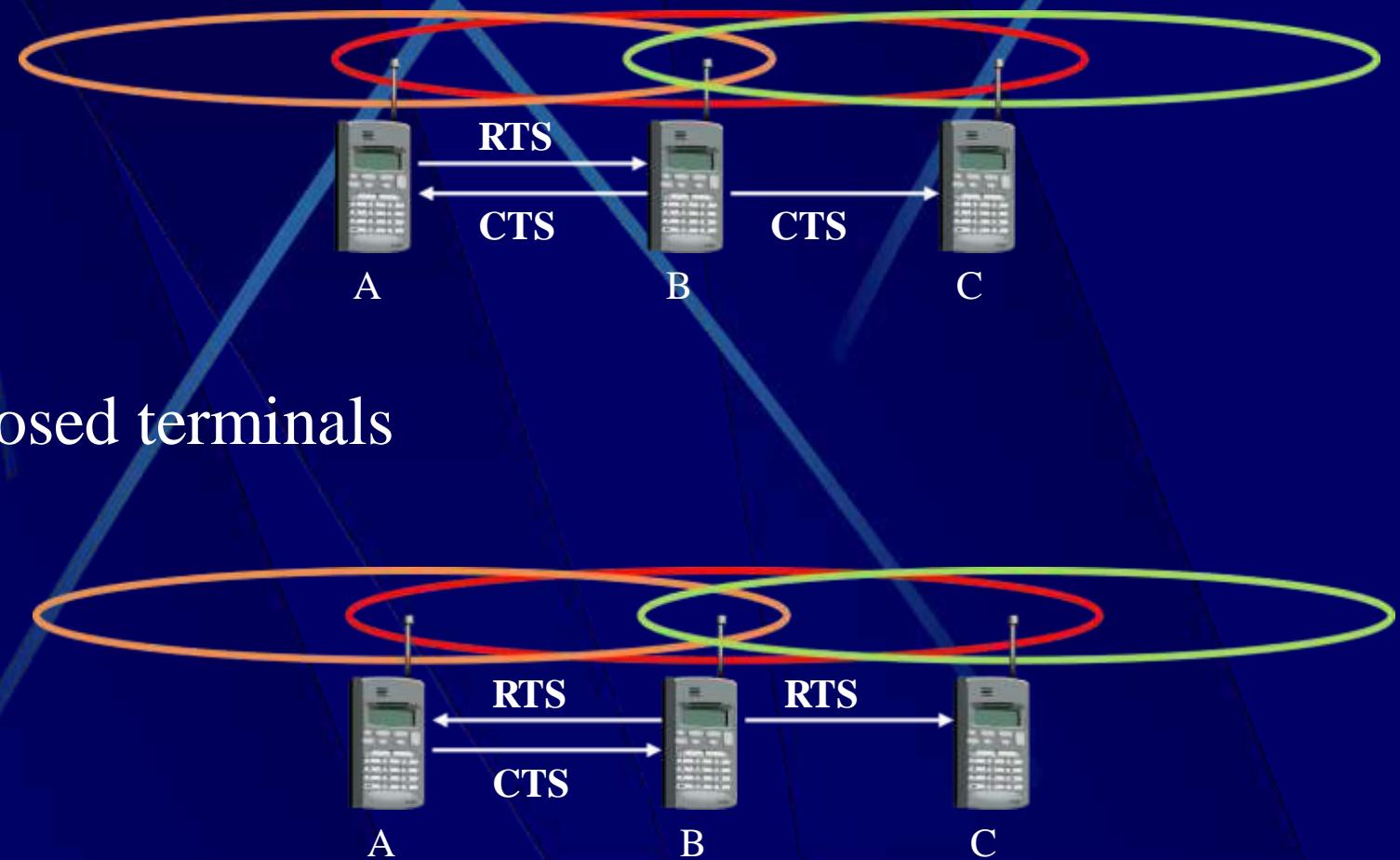
(b)

The MACA protocol.

- A sending an RTS to B.
- B responding with a CTS to A.

MACA examples

- ❖ MACA avoids the problem of hidden terminals
- ❖ A and C want to send to B
- ❖ A sends RTS first
- ❖ C waits after receiving CTS from B
- ❖ MACA avoids the problem of exposed terminals
- ❖ B wants to send to A, C to another terminal
- ❖ now C does not have to wait for it cannot receive CTS from A



MOBILE PLATFORMS AND APPLICATIONS

Mobile Device Operating Systems – Special Constraints & Requirements – Commercial Mobile Operating Systems – Software Development Kit: iOS, Android, BlackBerry, Windows Phone – M-Commerce – Structure – Pros & Cons – Mobile Payment System

Mobile Device Operating System Introduction

- Design and capabilities of a Mobile OS (Operating System) is very different than a general purpose OS running on desktop machines:
 - ❖ mobile devices have constraints and restrictions on their physical characteristic such as screen size, memory, processing power and etc.
 - ❖ Scarce availability of battery power
 - ❖ Limited amount of computing and communication capabilities
- Thus, they need different types of operating systems depending on the capabilities they support. e.g. a PDA OS is different from a Smartphone OS.
- Operating System is a piece of software responsible for management of operations, control, coordinate the use of the hardware among the various application programs, and sharing the resources of a device.

Operating System Structure

- A mobile OS is a software platform on top of which other programs called application programs, can run on mobile devices such as PDA, cellular phones, Smartphone and etc.



Mobile Operating System Platforms

□ There are many mobile operating systems. The followings demonstrate the most important ones:

- ❖ Java ME Platform
- ❖ Palm OS
- ❖ Symbian OS
- ❖ Linux OS
- ❖ Windows Mobile OS
- ❖ BlackBerry OS
- ❖ iPhone OS
- ❖ Google Android Platform

Special Constraints & Requirements

There are special constraints under which the operating system of a mobile device to operate

- ❖ Limited memory
- ❖ Limited screen size
- ❖ Miniature keyboard
- ❖ Limited processing power
- ❖ Limited battery power
- ❖ Limited and fluctuating of the wireless medium

Limited memory

- ❑ Mobile device has less permanent & volatile storage compared to desktop
- ❑ So OS must be small as possible but should provide rich set of functionalities to meet user needs

Limited screen size

- Size of mobile device needs to be small to make portable
- This limits size of display screen
- New innovative UI needs to be supported by mobile OS to overcome this constraint
- Eg: provide easy configurability to the interface to suit individual preferences, switching between menu & iconic interfaces etc

Miniature keyboard

- ❑ Provided with a small keyboard or small-sized display screen designed to be used as a keyboard
- ❑ Difficult to type documents and entering long commands

Limited processing power

- ❑ ARM based processor
- ❑ They are powerful & cheaper compared to desktop but slower

Limited battery power

- Due to the restrictions placed on their size & weight mobile device usually has a small battery
- In spite of the small battery a mobile phone is expected to support long talk time
- Mobile OS should minimize power consumption
- How?
 - by putting processor & display screen into sleep mode

Limited and fluctuating of the wireless medium

- ❑ Wireless medium directly affected by atmospheric noise
- ❑ So speed of communication(bandwidth) of a wireless channel may fluctuate due to mobility of handsets

Special service Requirements

- ❖ Support for specific communication protocols
- ❖ Support for a variety of input mechanism
- ❖ Compliance with open standard
- ❖ Extensive library support
- ❖ Support for integrated development environment(IDE)

Support for specific communication protocols

- ❑ Mobile devices required to be connected to the base station & various types of peripheral devices, computers & other mobile devices.
- ❑ So communication protocol support is required.
- ❑ These protocols depends upon generation(1G,2G etc)
- ❑ Eg: TCP/IP, Wireless protocol

Support for a variety of input mechanism

- ❖ Keyboard
- ❖ Touch screen

Compliance with open standard

- ❑ It helps in the development of innovative applications
- ❑ It reduces the cost of development & time to market by the mobile handset manufactures

Extensive library support

- ❑ Includes library support for
- ❑ Email
- ❑ SMS
- ❑ MMS
- ❑ Bluetooth
- ❑ Multimedia
- ❑ User interface primitives
- ❑ GSM/GPRS functionalities

Support for integrated development environment(IDE)

- Eclipse
- Android developer studio

Commercial Mobile Operating System

- Windows Mobile OS
- Palm OS
- Symbian OS
- iOS
- Android
- BlackBerry OS

Windows Mobile OS

- Windows Mobile is a compact operating system designed for mobile devices and based on Microsoft Win32.
- It is run on Pocket PCs, Smart phones and Portable media centers.
- It provides ultimate interoperability. Users with various requirements are able to manipulate their data.

Features:

- Graphics/Window/Event Manager(GWE) component handles all input & output
- Provides virtual memory management
- Support security through cryptography
- Does not provide true multitasking –when one application in the background goes into inactive and gets active only when it comes to foreground

Palm OS

- ❖ Palm OS is an embedded operating system designed for ease of use with a touch screen-based graphical user interface.
- ❖ It has been implemented on a wide variety of mobile devices such as smart phones, barcode readers, and GPS devices.
- ❖ It is run on Arm architecture-based processors. It is designed as a 32-bit architecture.

Palm OS Features

- The key features of Palm OS are:
 - 1)A single-tasking OS:
 - Only one application run at a time.
 - Eg: During voice communication we can not use calculator,read SMS etc

Palm OS Features (Cont.)

- 2) Memory Management:
 - To keep OS small & fast memory areas are not isolated.
 - Any misbehaving application can crash the system
- 3) Palm supplies Palm emulator
 - This helps to develop & debug Palm programs on a PC before running on Palm Hardware
- 4) It supports a handwriting recognition-based system for user input.

Palm OS Features (Cont.)

- 5) HotSync technology for synchronization with PC computers
- 6) Sound playback and record capabilities
- 7) TCP/IP network access
- 8) Support of serial port, USB, Infrared, Bluetooth and Wi-Fi connections
- 9) Defined standard data format for PIM (Personal Information Management) applications to store calendar, address, task and note entries, accessible by third-party applications
- 10) Security model:
- 11) Device can be locked by password, arbitrary application records can be made private

Symbian OS

- ❑ Symbian OS is 32 bit, little-endian operating system, running on different flavors of ARM architecture.
- ❑ It is a multitasking operating system and very less dependence on peripherals.
- ❑ Kernel runs in the privileged mode and exports its service to user applications via user libraries.

Symbian OS Features:

- Real time ,multitasking Pre-emptive 32-bit bit OS runs on ARM Processor
- Microkernel based OS
- 2 flavours of OS:
 - Series 60:
 - large size colour screen
 - easy to use interface
 - extensive suit of applications
 - mainly used on Nokia's smartphone & samsung handset
 - UIQ(user interface Quartz) interface:
 - software package developed by UIQ technology for symbian OS
 - it is a GUI layer helps third party application developers to develop applications with UI

Symbian OS Features(Ctd)

- Supports communication & networking protocols like TCP, UDP, FTP etc
- Supports pre-emptive multitasking scheduling & memory protection
- CPU switched into low power mode when application is not responding to an event.
- Optimized for low-power & memory requirements.
- Multimedia: it supports audio, video recording, playback and streaming, and Image conversion.
- Fully object-oriented design
- Carbide is an IDE available for C++ application development on symbian OS

iPhone OS

- ❖ iPhone OS is an operating system run on iPhone and iPod.
- ❖ It is based on Mac OS X(Mach Kernel).
- ❖ Fully owned & controlled by Apple & not design for other mobile phone vendors
- ❖ Apple does not license iOS for installation on third-party hardware
- ❖ **Features:**
- ❖ User interactions with OS include Gestures like swipe,tap,pinch & reverse pinch
- ❖ It has internal accelerometers used by some applications for shaking of the device as the undo command, rotating device to switch the display mode from portrait to landscape etc

iPhone OS

- ❖ Mac OS X has a preemptive multitasking environment.
- ❖ Preempting is the act of taking the control of operating system from one task and giving it to another task.
- ❖ It supports real-time behavior.
- ❖ In Mac OS X, each application has access to its own 4 GB address space.
- ❖ Not any application can directly modify the memory of the kernel. It has a strong mechanism for memory protection.

BlackBerry OS

- ❖ OS designed for BlackBerry smart phones
- ❖ It is a Proprietary OS(not free)
- ❖ Since it is not free details of its architecture is not been published.
- ❖ BlackBerry OS has a multitasking environment.
- ❖ It enables heavy use of input devices like trackball, and scroll wheel.
- ❖ It does not support touchpad.
- ❖ It is an event-driven Operating System.
- ❖ BlackBerry Smartphone's CPU architecture is based on ARM XScale. The other BlackBerry devices has Intel-based processors.
- ❖ Security: hardware based message encryption

Google Android Platform

- Google set up the Open Handset Alliance(OHA) in 2007
- OHA-group of companies & 82 technology
- It is a platform and an operating system for mobile devices based on the Linux operating system.
- It allows developers design applications in java like language using Google-developed java libraries.
- It supports a wide variety of connectivity such as GSM, WiFi, 3G, ...

Android Features

- Provide the ability to use either a phone-based keyboard or a touch screen
- Provide a built-in full web browser
- Open for third party developers
- Android SDK works in Eclipse environment
- Provides RDBMS SQLite for data storage & data sharing
- It has several innovative pre-installed applications like Gmail, Maps, Voice search etc

Android Software stack

- Various layers are:
 - Kernel
 - Libraries & Runtime
 - Application Framework
 - Application layer

Application Layer

- ❑ Set of applications
- ❑ Web browser, email client, SMS program, maps, calendar, contacts etc

Application framework

- Used to implement a standard structure for different applications
- Provides a set of services that an application programmer can make use of it.
- These services includes managers & content providers
- Content providers enables application to access data from other applications
- Notification manager allows an application to display custom alerts on the status bar.

Libraries & run time

- Libraries are written using languages like C & C++
- These are called through java interface-includes 2D & 3D graphics
MPEG-4,MP3,SQLite

Kernel

❑ Linux Kernel

TABLE 9.1 A Comparison of the Features of Three Popular Mobile Operating Systems

<i>Feature</i>	<i>Android</i>	<i>Symbian OS</i>	<i>Windows Phone 7</i>
License	Public, Free, and Open Source	Initially was private, later became public.	Proprietary
Footprint	250 KB	200 KB	300 KB
Change of UI	Possible	No	No
Power management	Yes	Yes	Yes
Kernel	Linux with minor changes	Proprietary	Win CE
True multitasking	Yes	Yes	No
Premptive scheduling	Yes	Yes	Yes
Demand paging	Yes	Yes	Yes
CPU architecture supported	ARM, MIPS, x 86	ARM	ARM

M-Commerce

Involves carrying out any activity related buying and selling of commodities, services or information using the mobile hand held devices.

- ❖ Applications of M-Commerce

M-commerce applications can be broadly categorized into B2C and B2B.

- Functionality
- Brand
- Packaging
- Services

Product

Price

Target
Market

Promotion

Place

- Advertising
- Sales force
- Publicity
- Sales promotion

- List Price
- Discounts
- Bundling
- Credit Terms

- Channel
- Inventory
- Logistics
- Distribution

Business-to-Consumer (B2C) Applications

- ❖ Advertising
- ❖ Comparison shopping
- ❖ Information about a product
- ❖ Mobile ticketing
- ❖ Loyalty and payment service
- ❖ Interactive advertisement
- ❖ Catalogue shopping

Business-to-Business (B2B) Applications

- ❖ Ordering and delivery confirmation
- ❖ Stock tracking and control
- ❖ Supply chain management
- ❖ Mobile inventory management

M-Commerce Structure

- ❖ Content provider implements an application by providing two sets of programs: Client-side and Server-Side
- ❖ Client side programs run on the browsers installed on users mobile.
- ❖ Server side programs performs database access and computations, resides on the host computers(Servers)

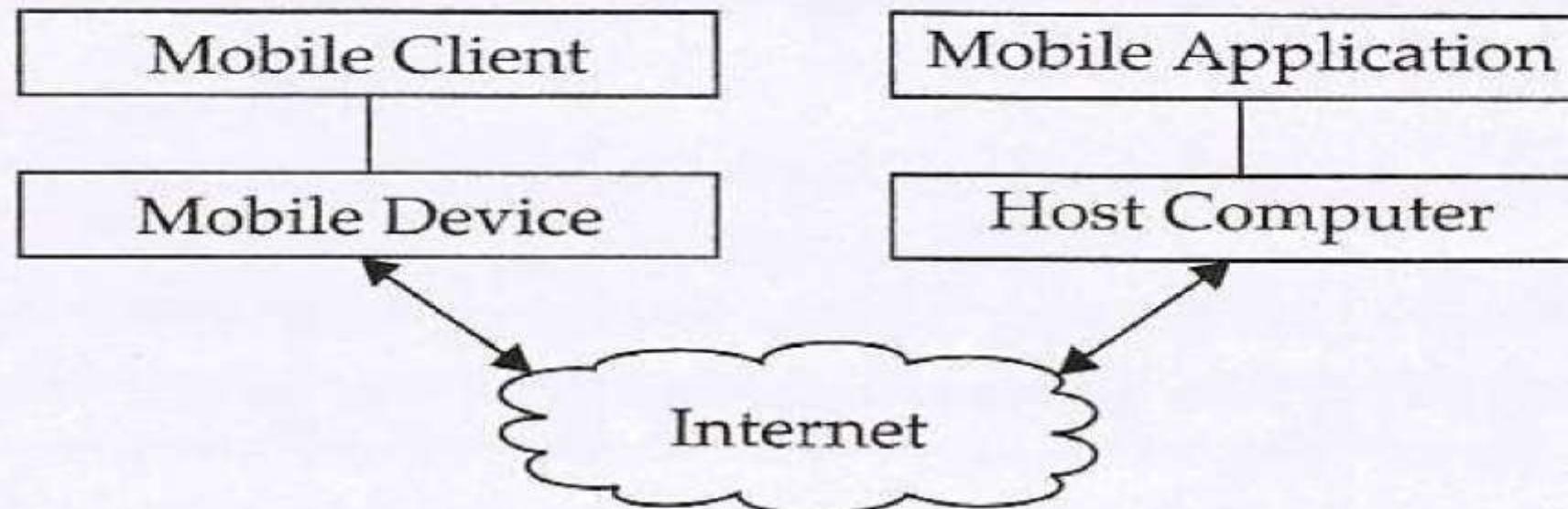


Figure 11.1 Architecture of a mobile commerce framework.

Mobile devices:

- Hand-held devices interfaced to mobile user, user specify their requests using interface programs,
- which are transmitted to mobile commerce application
- The result obtained from the mobile commerce application are displayed in suitable formats.

Mobile middleware

- The purpose of mobile middleware is to seamlessly and transparently map the internet content to mobile phones
- It also handle encrypting and decrypting communication for secure transaction.

Network

- The request are delivered to the closet wireless access point or base station or wired network such as internet for mobile commerce system

Host computers

- Process and stores all information needed for mobile commerce.
- It consists of three parts web servers, database servers and application program and support software.

M-Commerce Pros & Cons

Advantages

- For business organization: Customer convenience, cost savings and new business opportunities.
- For customer: Any where, any time shopping using light weight device.
- Without physically visiting to store identifying the right product at the lowest price.
- Highly personalized thereby providing an additional level of convenience to customer.

Disadvantages

- Mobile device not offer graphics or processing power of a PC-small screen
- The small screens of mobile devices limit the complexity of application.
- Network imposes several types of restriction.
- Security

Mobile Payment System

Mobile payment or m-payment defined as any payment instrument where a mobile device is used to initiate, authorize and confirm an exchange of financial value in return for goods and services.

OR

Mobile payment, also referred to as mobile money, mobile money transfer, and mobile wallet generally refer to payment services operated under financial regulation and performed from or via a mobile device.

Mobile Payment Schemes

Three popular types of M-payment schemes are currently used are

- I. Bank account based
- II. Credit card based
- III. Micropayment

- ❖ In each of these approach, a third party service provider (Bank, Credit card company or telecom company) make payment on the customer's behalf .
- ❖ The service provider may charge small amount as service charge

Bank account based M-payment

- The bank account of the customer is linked to his mobile number.
- When the customer makes an M-payment transaction with vendor, the bank account of the customer is debited and the value is credited to the vendor's account.

Credit card based M-payment

- The Credit card number is linked to mobile number of customer.
- When the customer makes an M-payment transaction with vendor, the credit card is charged and the value is credited to the vendor's account.

Micropayment

- ❖ The Micropayment is for small purchase such as from vending machines.
- ❖ A customer makes a call to the number of a service provider where the per call charge is equal to the cost of the vending item.
- ❖ The micropayment scheme is implemented thorough the cooperation of the mobile phone operator and a third party service provider.
- ❖ Eg:Vending beverages from coca-cola machines

Mobile Payment Characteristics

- Easy to use
- General Purpose
- Interoperability
- Trust
- Cost-should not impose a high overhead cost
- Swift ness-response time should be reasonable
- Global payments

Mobile Payment solutions

- SMS based payment
- POS(Point-of-sale) based payment
- Bar code based payment
- NFC(Near Field Communication) based payment
- Mobile wallet

Security Issues

- ❖ M commerce is anticipated to introduced new security and privacy risks.
- ❖ Users of mobile device can be difficult to trace because of roaming of the users.
- ❖ The mobile device go on-line and off-line frequently, thus attacks would be very difficult to trace.
- ❖ Another risk unique to the mobile devices is the risk of loss or theft.
- ❖ A major problem in this regard is lack of mechanism to authenticate a particular user.

1.0 Introduction to Android

What is Android?

- Mobile operating system based on [Linux kernel](#)
- User Interface for touch screens
- Used on [over 80%](#) of all smartphones
- Powers devices such as watches, TVs, and cars
- Over 2 Million Android apps in Google Play store
- Highly customizable for devices / by vendors
- Open source

Android user interaction

- Touch gestures: swiping, tapping, pinching
- Virtual keyboard for characters, numbers, and emoji
- Support for Bluetooth, USB controllers and peripherals

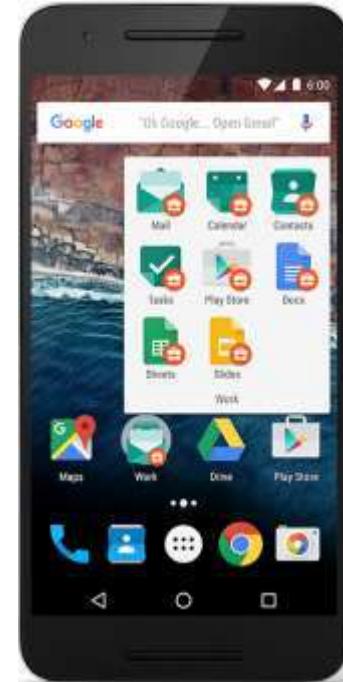
Android and sensors

Sensors can discover user action and respond

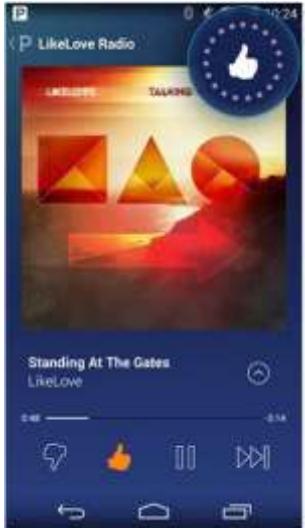
- Device contents rotate as needed
- Walking adjusts position on map
- Tilting steers a virtual car or controls a physical toy
- Moving too fast disables game interactions

Android home screen

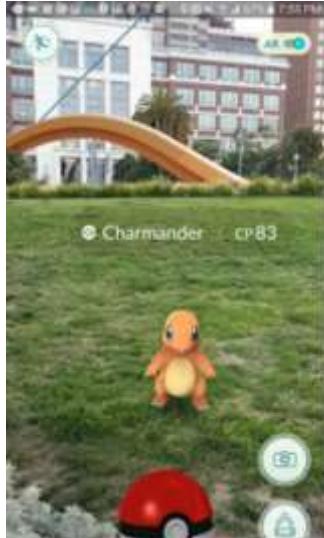
- Launcher icons for apps
- Self-updating widgets for live content
- Can be multiple pages
- Folders to organize apps
- "OK Google"



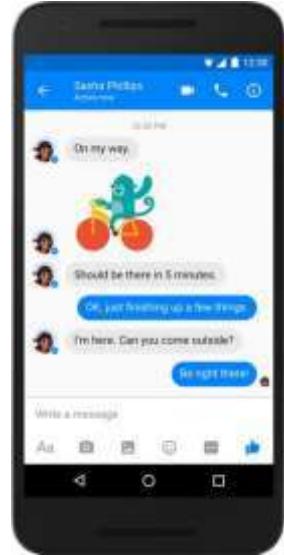
Android app examples



Pandora



Pokemon GO

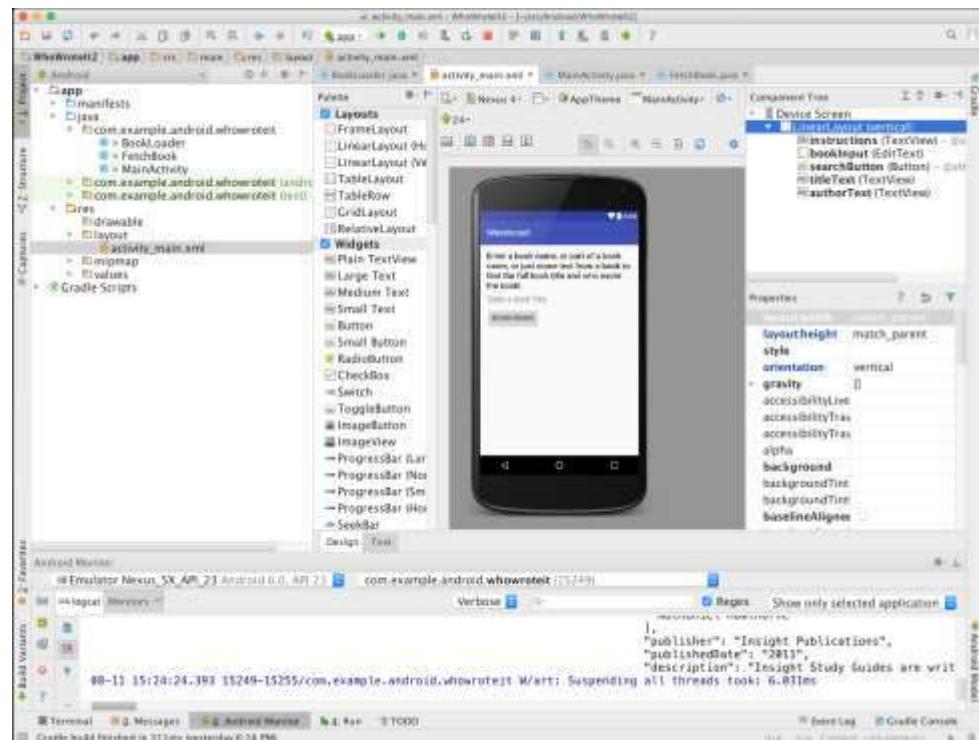


Facebook
Messenger

Android Software Developer Kit (SDK)

- Development tools (debugger, monitors, editors)
- Libraries (maps, wearables)
- Virtual devices (emulators)
- Documentation ([developers.android.com](https://developer.android.com))
- Sample code

Android Studio



- Official Android IDE
- Develop, run, debug, test, and package apps
- Monitors and performance tools
- Virtual devices
- Project views
- Visual layout editor

Google Play store

Publish apps through [Google Play](#) store:

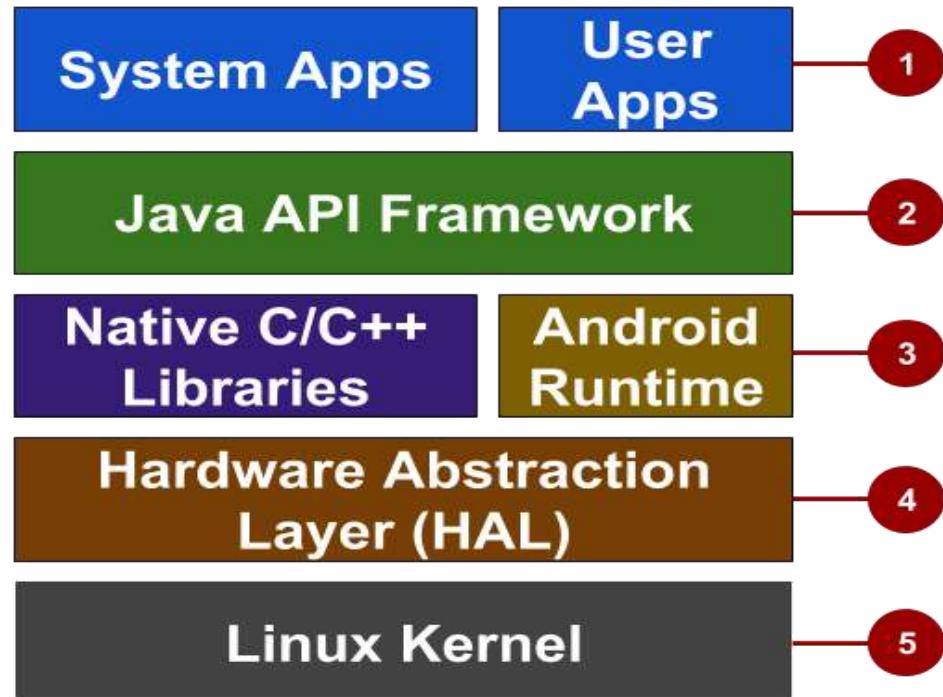
- Official app store for Android
- Digital distribution service operated by Google



Android Platform Architecture

Android Platform architecture

1. System and user apps
2. Android OS API in Java framework
3. Expose native APIs; run apps
4. Expose device hardware capabilities
5. Linux Kernel



System and user apps



- System apps have no special status
- System apps provide key capabilities to app developers

Example:

Your app can use a system app to deliver a SMS message.

Java API Framework

The entire feature-set of the Android OS is available to you through APIs written in the Java language.

- View class hierarchy to create UI screens
- Notification manager
- Activity manager for life cycles and navigation
- Content providers to access data from other apps

Android runtime

Each app runs in its own process with its own instance of the Android Runtime.

C/C++ libraries

- Core C/C++ Libraries give access to core native Android system components and services.

Hardware Abstraction Layer (HAL)

- Standard interfaces that expose device hardware capabilities as libraries

Examples: Camera, bluetooth module

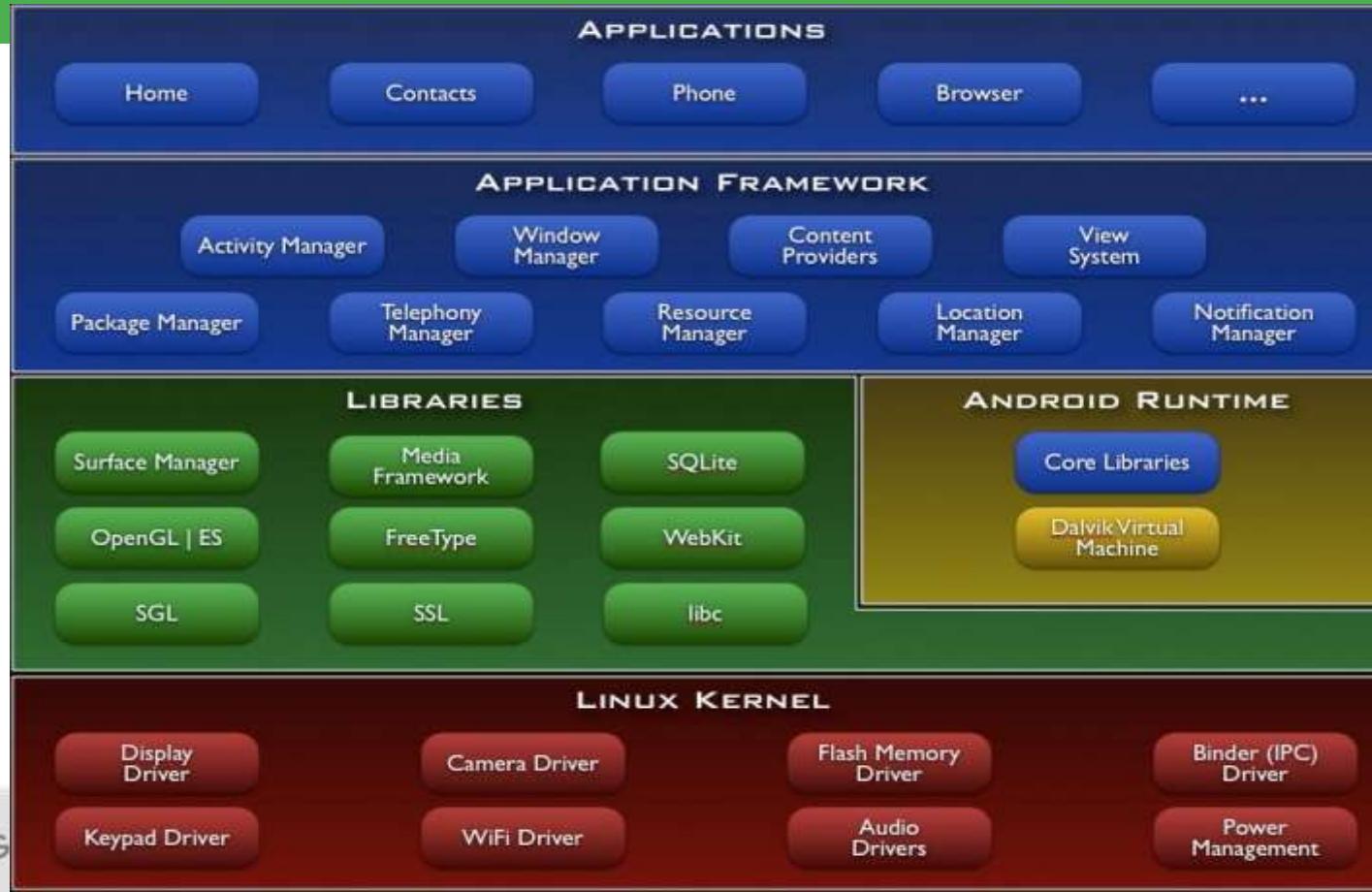
Linux Kernel

- Threading and low-level memory management
- Security features
- Drivers

Android Platform Architecture



Android Platform Architecture



Android versions



Codename	Version	Released	API Level
Honeycomb	3.0 - 3.2.6	Feb 2011	11 - 13
Ice Cream Sandwich	4.0 - 4.0.4	Oct 2011	14 - 15
Jelly Bean	4.1 - 4.3.1	July 2012	16 - 18
KitKat	4.4 - 4.4.4	Oct 2013	19 - 20
Lollipop	5.0 - 5.1.1	Nov 2014	21 - 22
Marshmallow	6.0 - 6.0.1	Oct 2015	23
Nougat	7.0	Sept 2016	24

[Android History](#) and
[Platform Versions](#)
for more and earlier
versions before 2011

ART(Android Run Time)

1. **Android Runtime (ART)** is an application runtime environment used by the Android operating system and some system services on Android.
2. ART is originally created specifically for the Android project.

ART Features

1. Ahead-of-time (AOT) compilation
2. Improved garbage collection
3. Development and debugging improvements

Ahead-of-time (AOT) compilation

1. which can improve app performance.
2. At install time, ART compiles apps using the on-device **dex2oat** tool.
3. This utility accepts [DEX](#) files as input and generates a compiled app executable for the target device.

Improved garbage collection

1. Garbage collection (GC) can impair an app's performance, resulting in choppy display, poor UI responsiveness, and other problems.
2. Garbage collection-cleaning up recently-allocated, short-lived objects.
3. GC to reduce background memory usage and fragmentation

Development and debugging

- **ART features**
- 1. ART offers a number of features to improve app development and debugging.
- 2. Support for more debugging features
- 3. Improved diagnostic detail in exceptions and crash reports

App behaviour on the Android runtime (ART)

1. The Android runtime (ART) is the default runtime for devices running Android 5.0 (API level 21) and higher. This runtime offers a number of features that improve performance and smoothness of the Android platform and apps

App behaviour on the Android runtime (ART)

1. Addressing garbage collection (GC) issues
2. Preventing JNI issues
3. Checking JNI code for garbage-collection issues
4. Error handling
5. Preventing stack size issues
6. Fixing AOT compilation issues
7. Reporting problems

Addressing garbage collection (GC)

- - 1. apps frequently find it useful to explicitly call [System.gc\(\)](#) to prompt garbage collection (GC).
 - 2. This should be far less necessary with ART
 - 3. A compacting garbage collector is under development in the [Android Open-Source Project \(AOSP\)](#) to improve memory management.
 - 4. Compact GC will move objects from memory.

Preventing JNI issues

1. If your app makes use of C/C++ code some JNI issues will come
2. Solution :

Checking JNI code for garbage-collection issues

Error handling

1. ART's JNI throws errors

Preventing stack size issues

1. There is no separate stack for native & JAVA code
2. ART has a unified stack for better locality.

Fixing AOT compilation issues

1. ART's Ahead-Of-Time (AOT) Java compilation should work for all standard Java code.
2. Compilation is performed by ART's dex2oat tool.

Reporting problems

1. If you run into any issues that aren't due to app JNI issues, report them via the Android Open Source Project Issue

App Development

What is an Android app?

- One or more interactive screens
- Written using Java Programming Language and XML
- Uses the Android Software Development Kit (SDK)
- Uses Android libraries and Android Application Framework
- Executed by Android Runtime Virtual machine (ART)

Challenges of Android development

- Multiple screen sizes and resolutions
- Performance: make your apps responsive and smooth
- Security: keep source code and user data safe
- Compatibility: run well on older platform versions
- Marketing: understand the market and your users
(Hint: It doesn't have to be expensive, but it can be.)

APP COMPONENTS

1. The basic components of any android application are the following:

- Activities
- Intent and broadcast receivers
- Services
- Content Providers
- Widgets and Notifications



Figure Principal Ingredients of android application

Activity

1. An activity is the first stepping stone in building an Android user application.
2. It provides the space to the user for doing anything and everything.
3. For example, opening a contact, dialing a caller, etc.
4. Everything is done by interacting with a window and that every window is provided by an activity.
5. A window is provided to each activity where user interfacing is done.
6. Generally, every Android application has more than one activity.
7. There is one “main” activity. All other activities are child activities

Activity Examples

1. An email app might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails.
2. A camera app can start the activity in the email app that composes new mail to allow the user to share a picture

Service

1. A service can be connected to a component which can even do inter process communication (IPC).
2. For example, when you receive your email updates in inbox it is a service.
3. You get the notification of new e-mail even if you are not using the e-mail app or doing something else.

Services Example

1. It is a component that runs in the background to perform long-running operations or to perform work for remote processes.
2. A service does not provide a user interface.
3. For example, a service might play music in the background while the user is in a different app,

Content Providers

1. A *content provider* manages a shared set of app data that you can store in the file system, in a SQLite database, on the web, or on any other persistent storage location that your app can access
2. Through the content provider, other apps can query or modify the data if the content provider allows it.
3. For example, the Android system provides a content provider that manages the user's contact information.
4. Content providers as the name indicates provides content of one process to another hence it acts as an interface.

Android Intents & Broadcast Receivers

1. **Android Intents** are the communication medium .i.e., app components send messages to one another like you do with your friends.
2. It is a messaging object. It can be used to query an action from another app component.
3. Android Intent can be used to instantiate a new activity or get result from another activity.
4. A service can be started by passing intent to perform a single operation. A broadcast can be sent to other apps by passing intents.

Broadcast

1. There are two types of broadcasts:

1. Normal Broadcasts:

- These are asynchronous in nature.
- Many receivers can be activated at the same time which doesn't have any defined order.
- But they are very efficient.

2. Ordered Broadcasts:

1. They are synchronous in nature.
2. Broadcast received by one receiver passes it to other receivers.
3. Broadcasts are delivered to receiver on one-to-one and sequential basis.
Either receiver will pass result to another receiver or it may completely destroy the broadcast

Broadcast

1. Android **Broadcast** is a message which spreads out when any event occurs.
2. They are received by apps.
3. Android Intents can be used to deliver broadcasts to other apps.
4. For example, when your device boots up or switched on system generates a broadcast to all apps.
5. There should be a procedure or should be something which can receive these broadcasts.
6. These receptors are called broadcast receivers.

Broadcast Example

1. Many broadcasts originate from the system:
2. Eg:A broadcast announcing that the screen has turned off, the battery is low, or a picture was captured.
3. Apps can also initiate broadcasts—
4. for example, to let other apps know that some data has been downloaded to the device and is available for them to use.

Android Widgets and Notifications

1. **Android App widgets** are the small application views.
2. These views can be embedded into other applications.
3. They can receive updates on periodic basis.
4. what is a widget ?
5. A **widget** is a quick view of your app's functionality and data.
6. This view is accessible from home screen of your device.

widgets are of following types:

- **1. Informational Widget:** These Android widgets are going to display only that information to user which is important and dynamic in nature.

Example the information displayed on your home screen saying time and weather condition is a widget of this type.

1. **Collection Widgets:**
2. These Android widgets scroll in top-to-down direction.
Collection of information of same type and then enabling user to open any one of them to full detail.
3. Example is your e-mail app which will display all the mails in your inbox and then allow you to open any one of them.

- 1. Control Widgets:** Displays the most frequently used functionalities which user might want to control from home screen.
2. For example in a video app, you can pause, play, stop, go to previous track, move to next track is an example of control widget.

1. • Hybrid Widgets:
2. These Android widgets combine features of all of the above three:

1. Notification

2. As the name says keeps the user aware of events going on.
3. User is kept informed like any news channel.
4. For e.g, everyone of us know about facebook or whatsapp, now notification system of app is responsible for informing you about any new friend request, chat request, or a new message from say, dvs or xyz, etc.

Component types

- **Activity** is a single screen with a user interface
- **Service** performs long-running tasks in background
- **Content provider** manages shared set of data
- **Broadcast receiver** responds to system-wide announcements

Think of Android as a hotel

- Your app is the guest
- The Android System is the hotel manager
- Services are available when you request them (intents)
 - In the foreground (activities) such as registration
 - In the background (services) such as laundry
- Calls you when a package has arrived (broadcast receiver)
- Access the city's tour companies (content provider)

Android application framework



1. • **Activity manager**:-It manages the lifecycle of applications. It enable proper management of all the activities. All the activities are controlled by activity manager.
2. • **Resource manager**:-It provides access to non-code resources such as graphics etc.
3. • **Notification manager**:-It enables all applications to display custom alerts in status bar.
4. • **Location manager**:- It fires alerts when user enters or leaves a specified geographical location.
5. • **Package manager**:-It is use to retrieve the data about installed packages on device.

1. • **Window manager**:-It is use to create views and layouts.
2. • **Telephony manager**:-It is use to handle settings of network connection and all information about services on device.

Applications



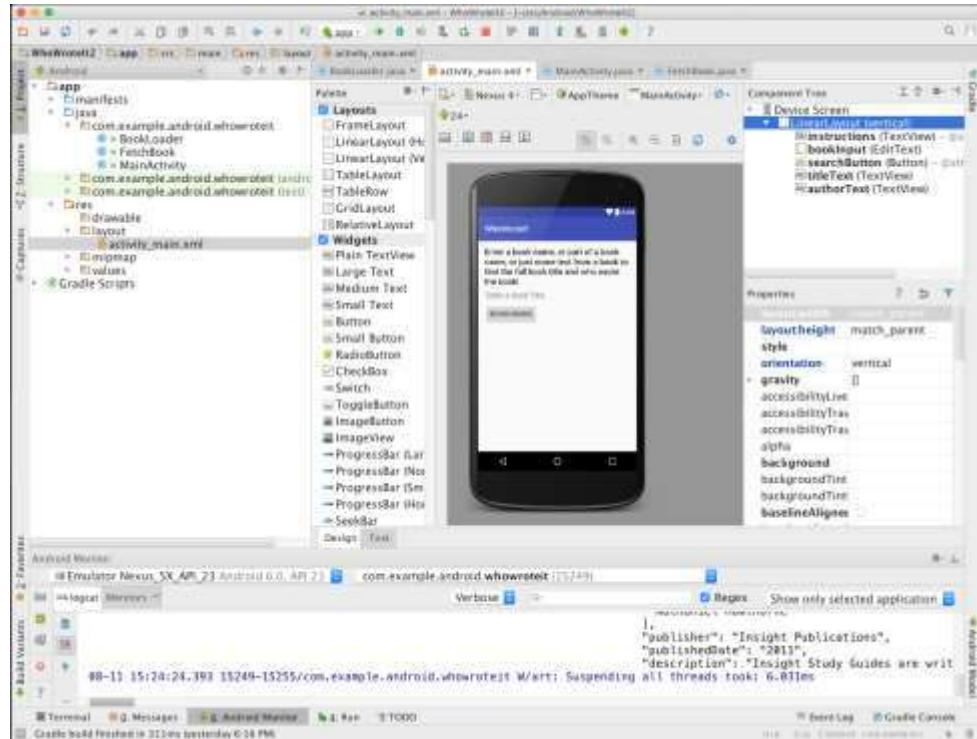
Applications Ctd..

1. You will find all the Android application at the top layer.
2. You will write your application to be installed on this layer only.
3. Examples of such applications are

Contacts Books, Browser, Games etc.

Android Studio

What is Android Studio?



- Android IDE
- Project structure
- Templates
- Layout Editor
- Testing tools
- Gradle-based build
- Log Console
- Debugger
- Monitors
- Emulators

This work is licensed under a [Creative Commons Attribution-NonCommercial 4.0 International License](https://creativecommons.org/licenses/by-nc/4.0/)

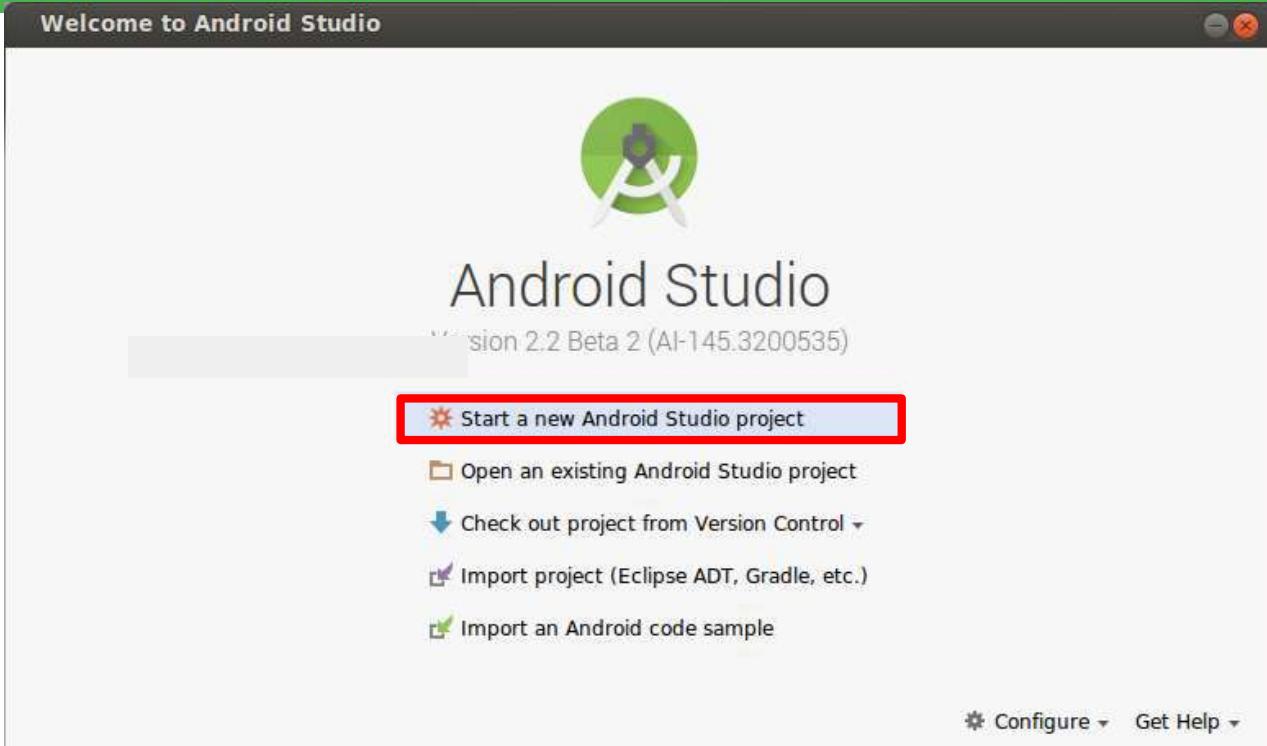


Installation Overview

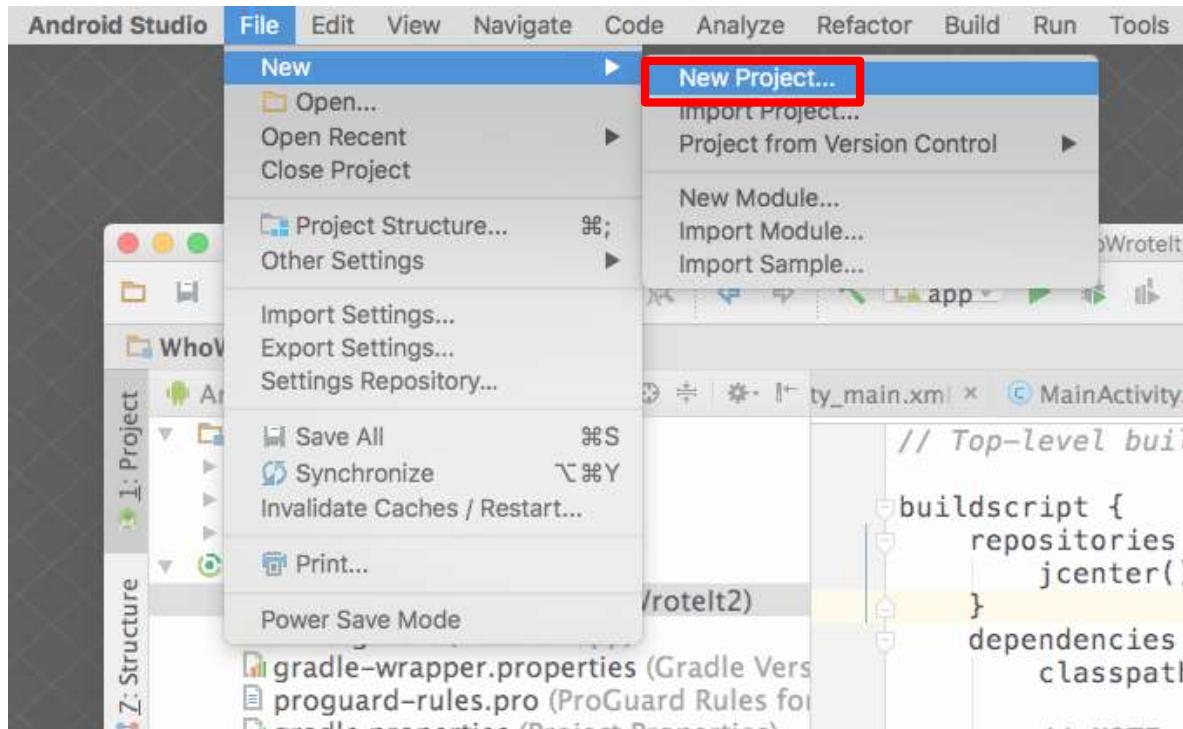
- Mac, Windows, or Linux
- Requires Java Development Kit (JDK) 1.7 or better from [Oracle Java SE downloads page](#)
- Set JAVA_HOME to JDK installation location
- Download and install Android Studio from [http://developer.android.com/sdk/index.html](#)
- See [1.1 P Install Android Studio for details](#)

Creating Your First Android App

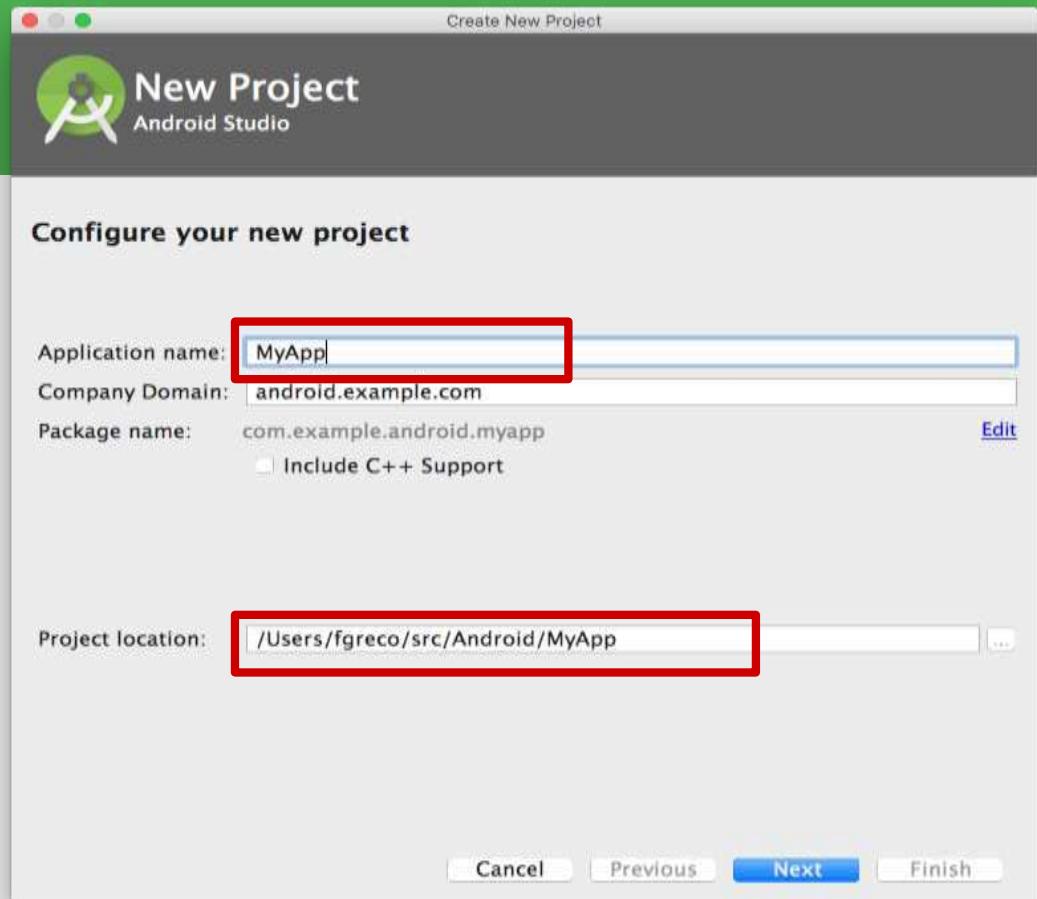
Start Android Studio



Create a project inside Android Studio



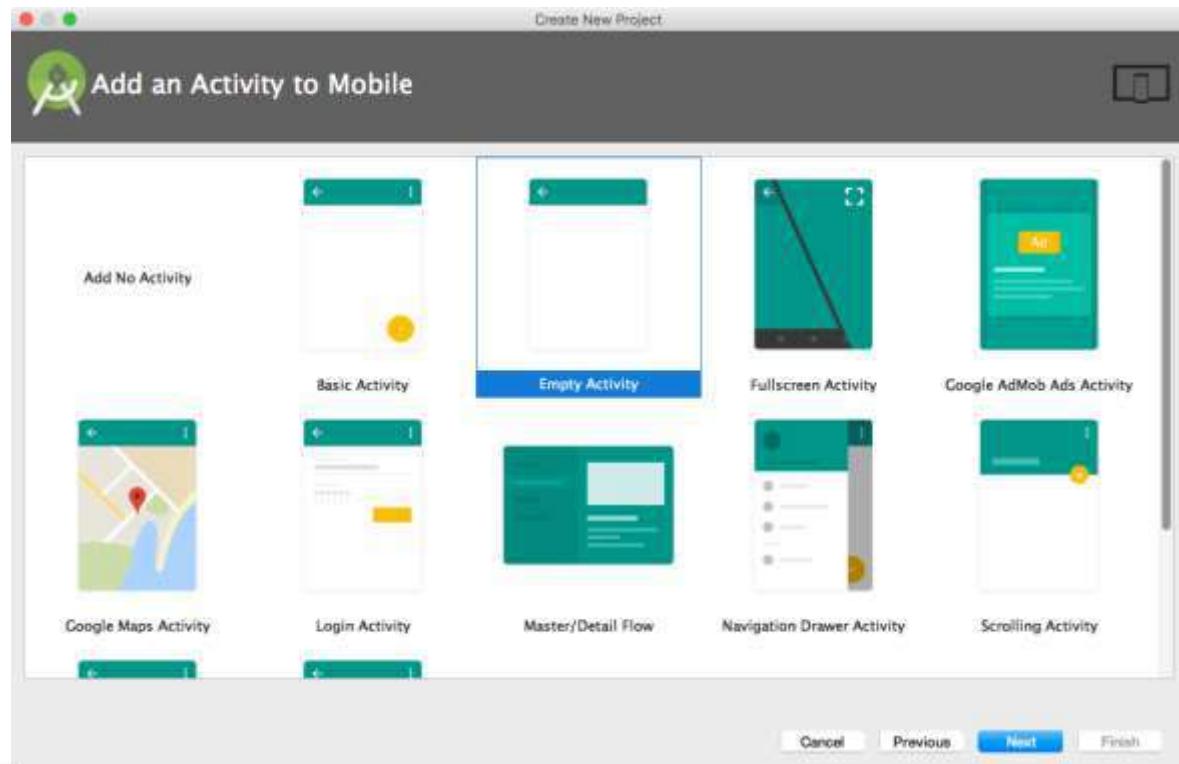
Name your app



Pick activity template

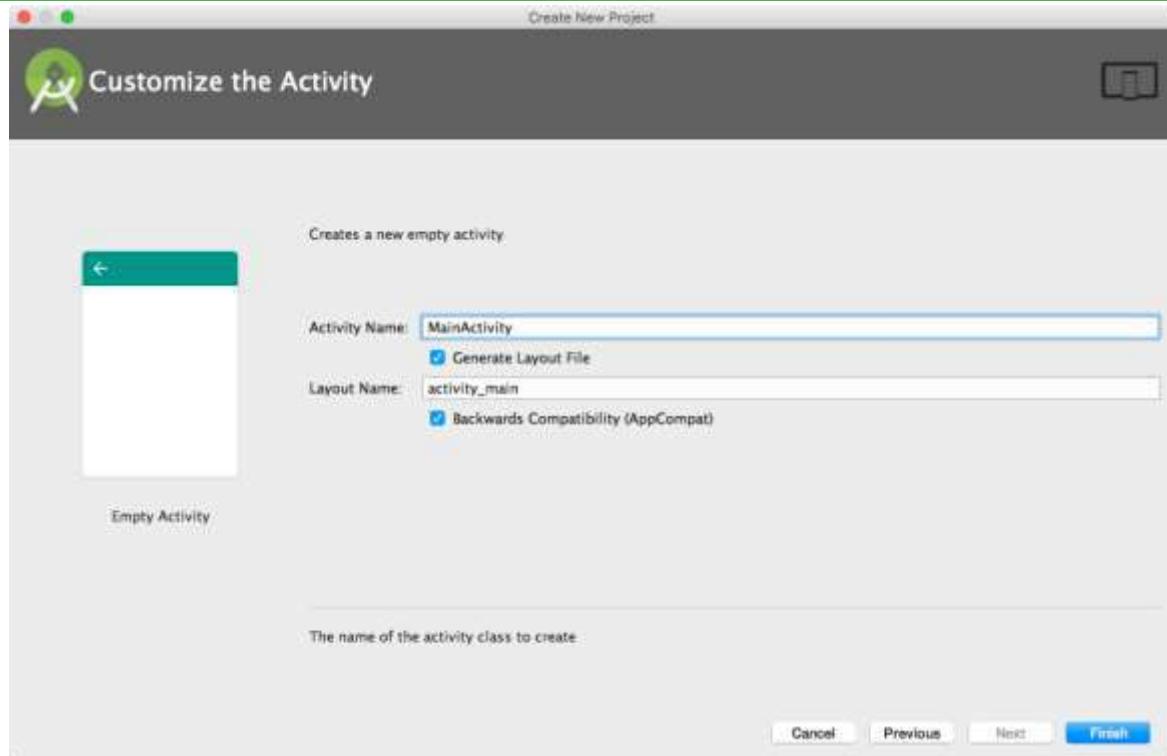
Choose templates for common activities, such as maps or navigation drawers.

Pick Empty Activity or Basic Activity for simple and custom activities.



Name your activity

- Good practice to name main activity `MainActivity` and `activity_main` layout
- Use AppCompat
- Generating layout file is convenient



Android Studio Panes

The screenshot shows the Android Studio interface with several panes:

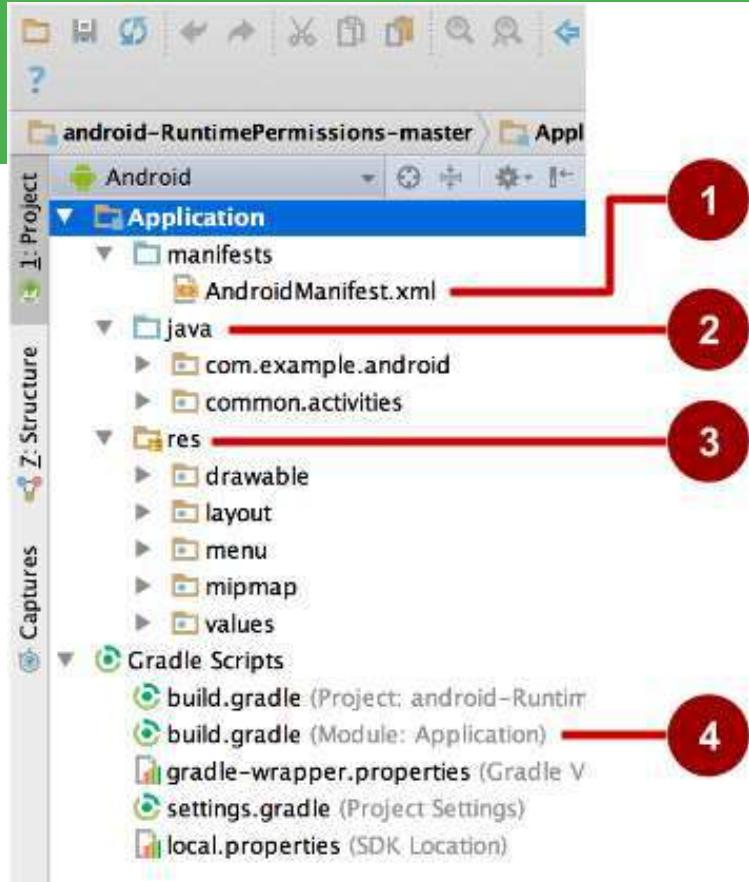
- Project** pane (left): Shows the project structure with modules like app, Java files (MainActivity), and resource files (layout, mipmap, values).
- Layout Editor** pane (top right): Displays the UI design for activity_main.xml. It includes a component tree showing a Linear Layout containing a button and a TextView labeled "show_count". The TextView has a large blue "0" displayed on it.
- Properties** pane (right side of Layout Editor): Shows properties for the "show_count" TextView, including ID, layout_width, layout_height, and text set to "@string/count_init".
- Android Monitor** pane (bottom right): Shows logcat output for the emulator. The log includes messages like "09-26 16:29:17.556 D/Emulator: [2724]" and "09-26 16:29:17.642 I/Emulator: connection established 0x7f06f9021c00, tid 1555".

Three specific sections are highlighted with green boxes:

- Project Files**: Points to the Project pane.
- Layout Editor**: Points to the Layout Editor pane.
- Android Monitors: logcat: log messages**: Points to the Android Monitor pane.

Project folders

- 1. manifests**—Android Manifest file - description of app read by the Android runtime
- 2. java**—Java source code packages
- 3. res**—Resources (XML) - layout, strings, images, dimensions, colors...
- 4. build.gradle**—Gradle build files

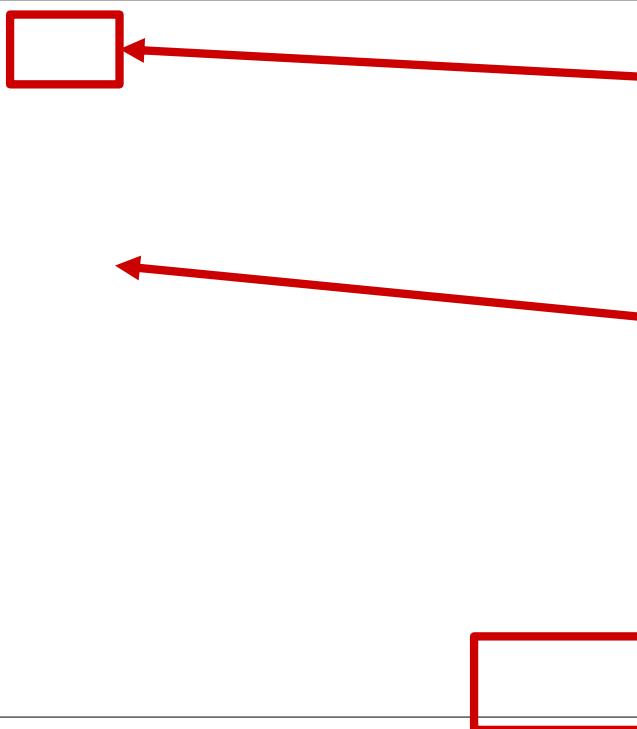


Gradle build system

- Modern build subsystem in Android Studio
- Three build.gradle:
 - project
 - module
 - settings
- Typically not necessary to know low-level Gradle details
- Learn more about gradle at <https://gradle.org/>

Run your app

The image part with relationship ID r1d1 was not found in the file.



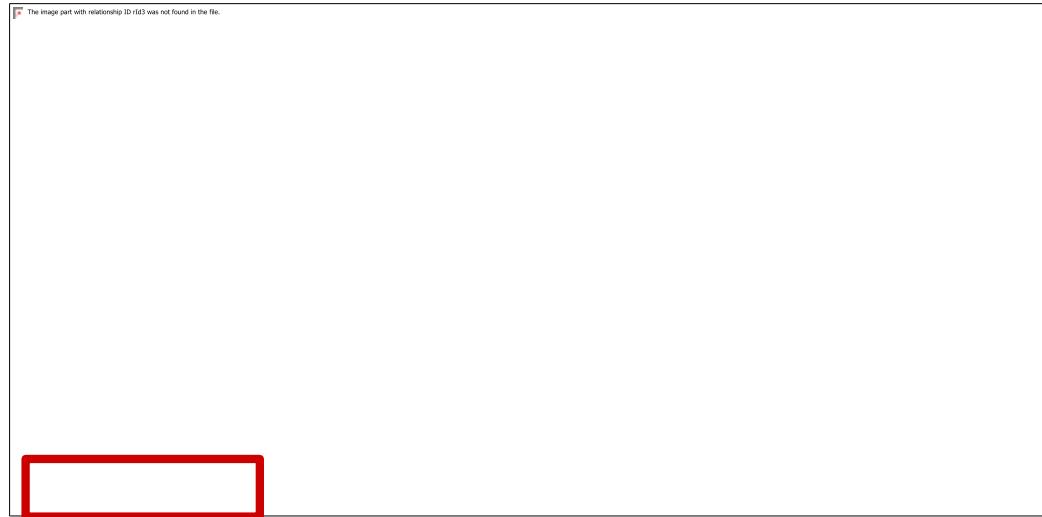
1. Run
2. Select virtual or physical device
3. OK

Create a virtual device

Use emulators to test app on different versions of Android and form factors.

Tools > Android > AVD Manager

or:



Configure virtual device

1. Choose hardware

2. Select Android Version

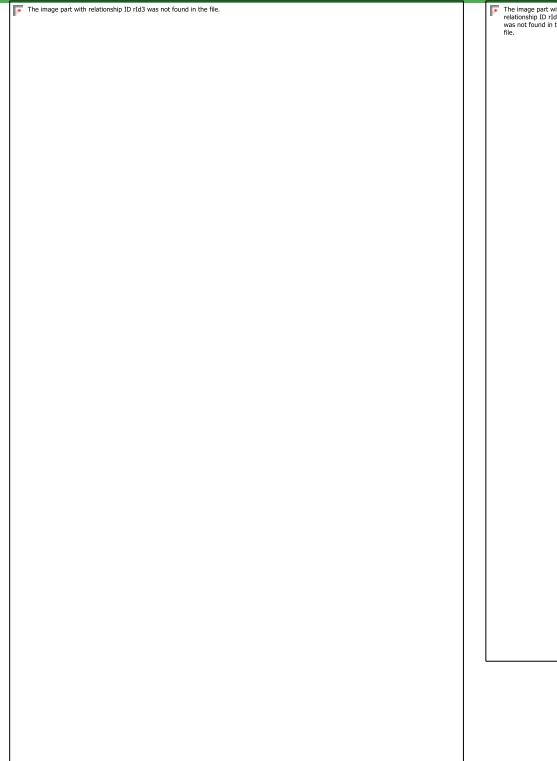
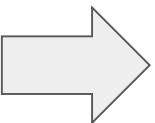
3. Finalize

The image part with relationship ID rId3 was not found in the file.

The image part with relationship ID rId4 was not found in the file.

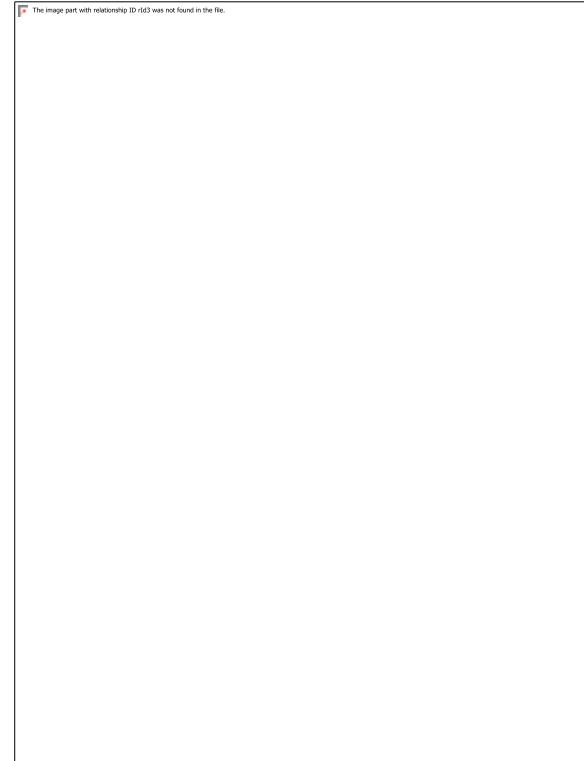
The image part with relationship ID rId5 was not found in the file.

Run on a virtual device



Run on a physical device

1. Turn on Developer Options:
 - a. **Settings > About phone**
 - b. Tap **Build number** seven times
2. Turn on USB Debugging
 - a. **Settings > Developer Options > USB Debugging**
3. Connect phone to computer with cable



Windows/Linux additional setup:

- [Using Hardware Devices](#)

Windows drivers:

- [OEM USB Drivers](#)

1. “ECLIPSE”
2. With a single download, the Eclipse ADT bundle
3. includes everything you need to begin developing
4. apps:
5. ☐ Eclipse + ADT plug-in
6. ☐ Android SDK Tools
7. ☐ Android Platform-tools
8. ☐ A version of the Android platform
9. ☐ A version of the Android system image for the emulator

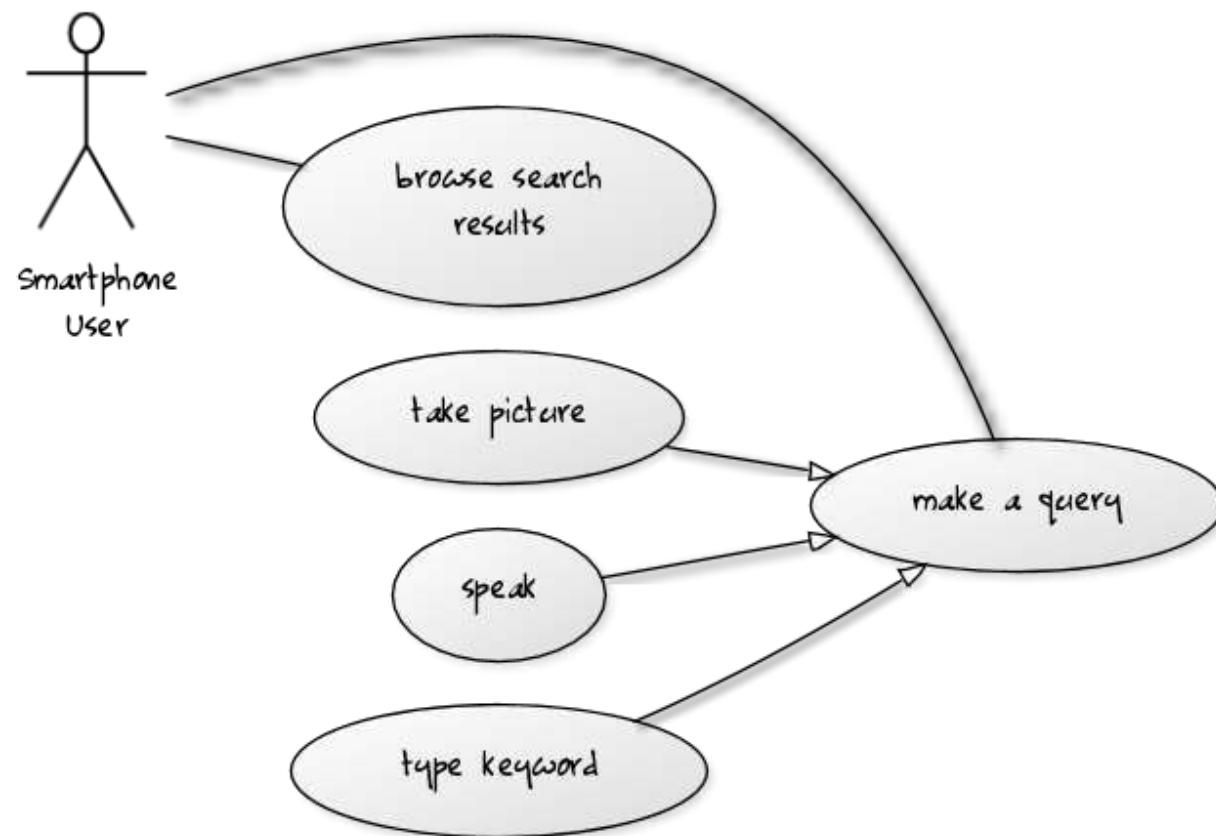
Android Application Use Case

Use case *Diagram (UML)*

UML DIAGRAM

- **UML deployment diagram** shows deployment of an application to Android.

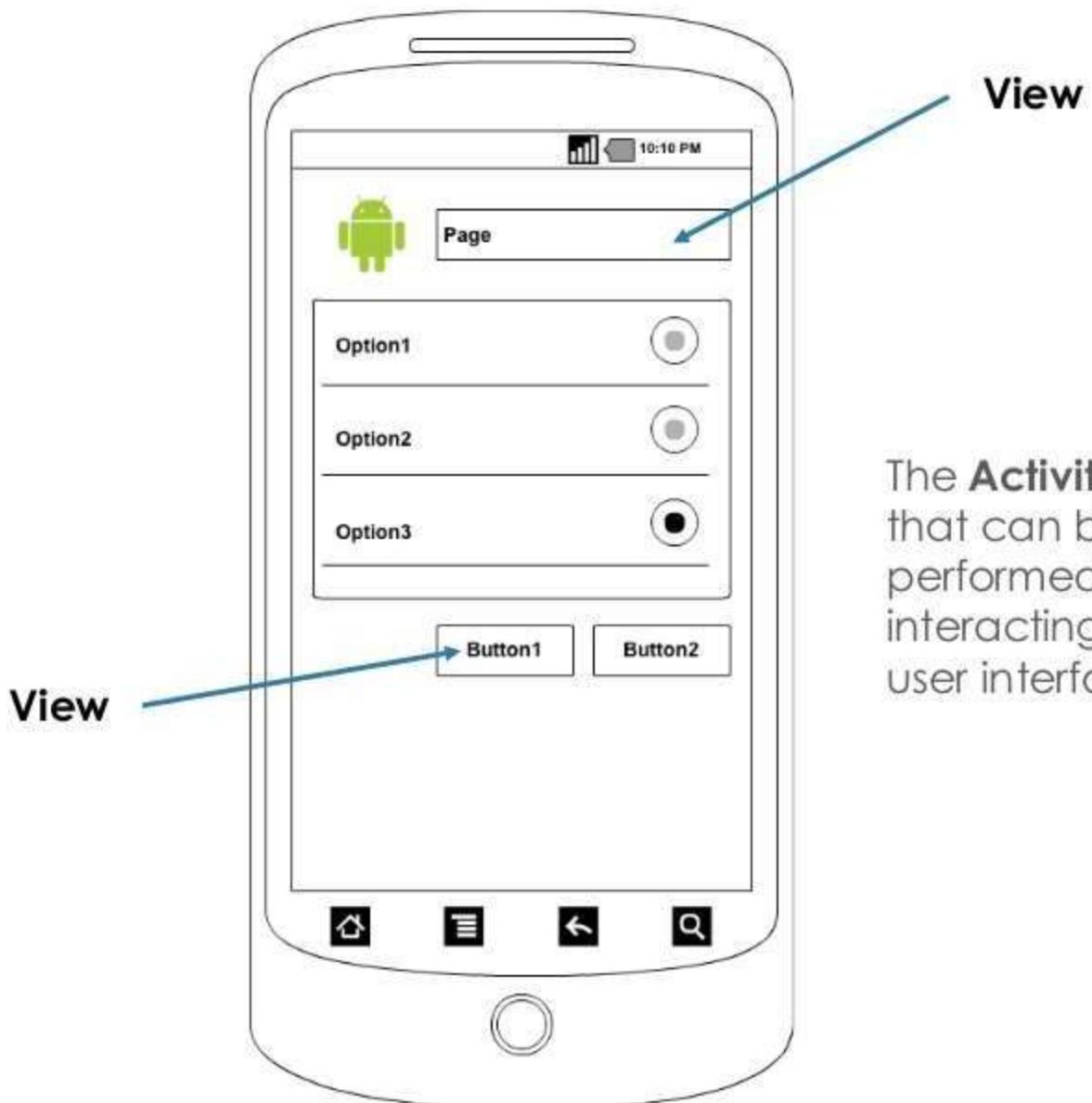
UML diagram



Activities and Views

- The **activity** is a single, focused thing that the user can do
 - Each activity is associated with a window in which to draw the user interface
- The **view** is the basic building block for user interface components
 - Responsible for drawing and event handling
 - Examples: button, textbox

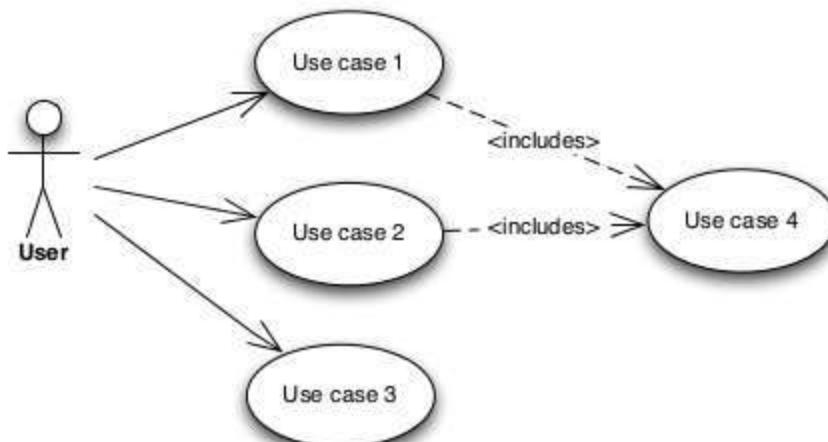
Application building blocks: UI components



The **Activity** is the task that can be performed by interacting with the user interface

Activities and use case diagrams

- Roughly, each activity corresponds to a use case
- As such, the use case diagram could also serve as the graph of connected activities



Resources

- Resource is an important part in Android apps.
- Stored in subfolders of **/res**
- There are **8 types of resources**:
- **anim/**: contains xml files that define property animations.
 - From code, we can access by R.anim
- **color/**: contains xmls files that define a state list of colors.
 - From code, we can access by R.color
- **drawable/** contains image files or xml files
 - From code ,we can access by R.drawable
- **layout/** contains xml files that define user interfaces.
 - From code,we can access by R.layout

Resource Overview (cont)

- **menu/** contains xml files that define app menus
Eg: Option Menu, Context Menu or Sub Menu.
From code, we can access by **R.menu**
- **raw/** contains arbitrary files in their raw form.
- You need to call **Resources.openRawResource()** with resource ID, which is **R.raw.filename** to open such raw files
- **xml/** contains arbitrary xml files that can be read at runtime by calling **Resources().getXML()**

Values

- **values/** contains xml files storing simple values such as string,integer, color.

There are some files in this folder:

- **arrays.xml** for resource arrays, and accessed from R.array
- **Integers.xml** for resource integer and accessed from R.integer
- **bools.xml** for resource boolean and accessed from R.bool
- **colors.xml** for color values and accessed from R.color
- **dimens.xml** for dimen values and accessed from R.dim
- **string.xml** for string values and accessed from R.string
- **styles.xml** for styles and accessed from R.style

AndroidManifest.xml

- Each Android project includes a manifest file, **AndroidManifest.xml**, stored in the root of its project hierarchy.
- The manifest **defines the structure** and **metadata** of your application, its components, and its requirements.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.myorg.twitterfeeds"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".TwitterFeedsDemoActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

AndroidManifest.xml

- Includes nodes for each of the **Activities**, **Services**, **Content Providers**, and **Broadcast Receivers** that make up your application and, using **Intent Filters** and **Permissions**, determines **how they interact** with each other and with other applications.
- **Specifies application metadata** (such as its icon, version number, or theme), and additional top-level nodes **can specify any required permissions** and **define hardware, screen, or platform requirements**
- The manifest is made up of a **root manifest tag** with a package attribute set to the project's package. It should also include an **xmlns: android attribute** that supplies several system attributes used within the file.

AndroidManifest.xml

- `versionCode`: Define the **current application version as an integer** that increases with each version iteration
- `versionName`: Specify a **public version** that will be displayed to users
- `installLocation`: Specify whether to allow (or prefer) for your application be **installed on external storage** (usually an SD card)
[`preferExternal | auto`]

AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.paad.myapp"
    android:versionCode="1"
    android:versionName="0.9 Beta"
    android:installLocation="preferExternal">
    [ ... manifest nodes ... ]
</manifest>
```

- **uses-sdk:** Enables **to define a minimum and maximum SDK version** that must be available on a device for your application to function properly, and **target SDK** for which it has been designed.
[**minSdkVersion**, **maxSdkVersion**, and **targetSdkVersion**].
 - The minimum SDK version **specifies the lowest version of the SDK**. If not specified a minimum version, **it defaults to 1, and your application crashes when it attempts to access unavailable APIs.**
 - The target SDK version attribute **enables you to specify the platform against which you did your development and testing**. Good practice is to update the target SDK of your application to the latest platform release after you confirm it behaves as expected, even if you aren't making use of any new APIs.
 - **maxSdkVersion: Unnecessary**

AndroidManifest.xml

- **uses-configuration:** Specifies each combination of input mechanisms are supported by your application. Normally no need to include this node, though it can be useful for games that require particular input controls.
 - `reqFiveWayNav` – Specify true for this attribute if you require an input device capable of navigating up, down, left, and right and of clicking the current selection. This includes both trackballs and directional pads (D-pads).
 - `reqHardKeyboard` – If your application requires a hardware keyboard, specify true.
 - `reqKeyboardType` – Lets you specify the keyboard type as one of nokeys, qwerty, twelvekey, or undefined.
 - `reqNavigation` – Specify the attribute value as one of nonav, dpad, trackball, wheel, or undefined as a required navigation device.
 - `reqTouchScreen` – Select one of notouch, stylus, finger, or undefined to specify the required touchscreen input.

```
<uses-configuration  
    android:reqFiveWayNav="true"  
    android:reqHardKeyboard="true"  
    android:reqKeyboardType="qwerty"  
    android:reqNavigation="trackball"  
    android:reqTouchScreen="finger" />
```

AndroidManifest.xml

- **uses-feature** – Use multiple uses-feature nodes **to specify which hardware features your application requires**. This prevents your application from being installed on a device that does not include a required piece of hardware, such as NFC hardware, as follows:

```
<uses-feature android:name="android.hardware.nfc" />
```

You can require support for any hardware that is optional on a compatible device. Currently, optional hardware features include the following:

- Bluetooth
- Camera
- Location
- Microphone
- NFC
- Sensors
- Telephony
- Touchscreen
- USB
- Wi-Fi

```
<uses-feature android:name="android.hardware.camera" />
<uses-feature
    android:name="android.hardware.camera.autofocus"
    android:required="false" />
<uses-feature
    android:name="android.hardware.camera.flash"
    android:required="false" />
```

<http://developer.android.com/guide/topics/manifest/uses-feature-element.html#features-reference>.

AndroidManifest.xml

- `supports-screens` – Enables you to **specify the screen sizes your application has been designed and tested to**. On devices with supported screens, your application is laid out normally using the scaling properties associated with the layout files you supply.
 - `smallScreens` – typically, QVGA screens
 - `normalScreens` – HVGA, including WVGA and WQVGA.
 - `largeScreens` – Screens larger than normal.
 - `xlargeScreens` – Typically tablet devices.
- Honeycomb MR2 (API level 13) introduced additional attributes:
 - `requiresSmallestWidthDp`
 - `compatibleWidthLimitDp`
 - `largestWidthLimitDp`

```
<supports-screens  
    android:largeScreens="true"  
    android:normalScreens="true"  
    android:smallScreens="false"  
    android:xlargeScreens="true" />
```

AndroidManifest.xml

- uses-permission – (Security model), **declare the user permissions your application requires**. Each permission you specify will be presented to the user before the application is installed. Permissions are required for many APIs and method calls, generally those with an associated cost or security implication (such as dialing, receiving SMS, or using the location-based services).

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

- permission – Your application components can also **create permissions to restrict access to shared application components**.
 - Your application components can then create permissions by adding an **android:permission** attribute. Then you can include a uses-permission tag in your manifest to use these protected components, both in the application that includes the protected component and any other application that wants to use it.

```
<permission android:name="com.paad.DETONATE_DEVICE"  
          android:protectionLevel="dangerous"  
          android:label="Self Destruct"  
          android:description="@string/detonate_description">  
</permission>
```

AndroidManifest.xml

- **application** – A manifest **can contain only one** application node.
 - During development if you include a **debuggable** attribute set to true to enable debugging, then be sure to disable it for your release builds.
 - The application node also acts as a container for the Activity, Service, Content Provider, and Broadcast Receiver nodes that specify the application components. You specify the name of your custom application class using the **android:name** attribute.

```
<application android:icon="@drawable/icon"
            android:logo="@drawable/logo"
            android:theme="@android:style/Theme.Light"
            android:name=".MyApplicationClass"
            android:debuggable="true">
            [ ... application nodes ... ]
</application>
```

- activity
- service
- provider
- receiver
- uses-library

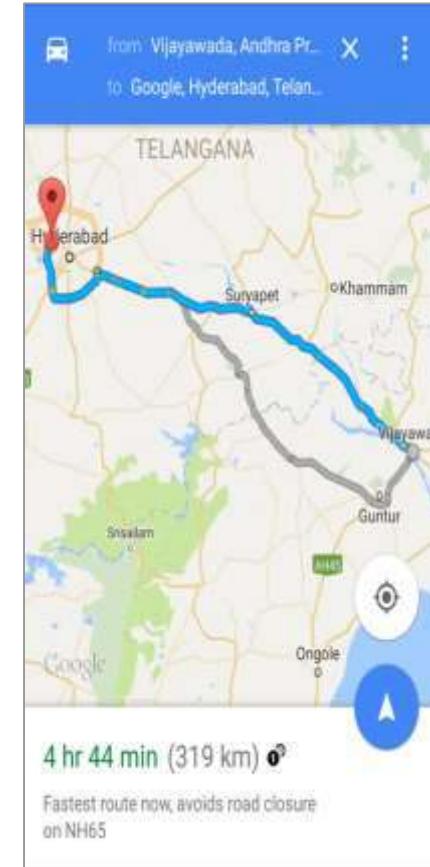
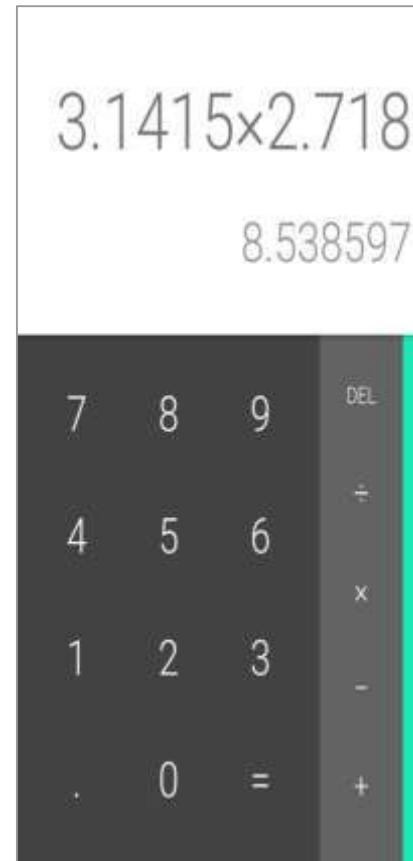
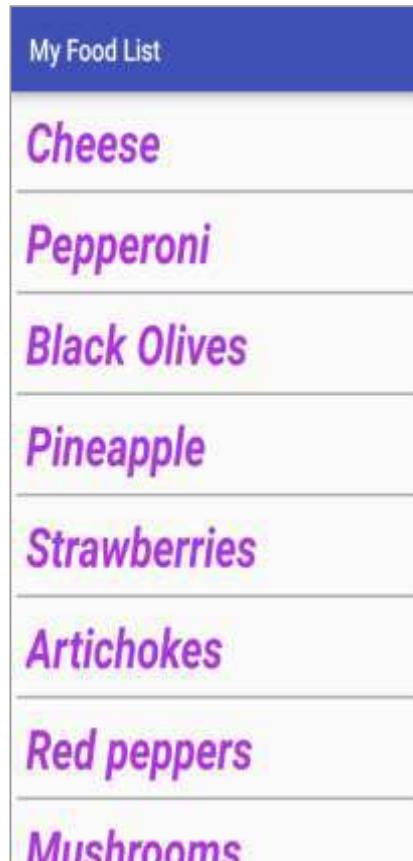
ACTIVITY

- An Activity is an application component that provides a screen with which users can interact in order to do something
- Eg: dial the phone, take a photo, send an email, or view a map.
- Each activity is given a window in which to draw its user interface.
- An Activity is an application component
- Java class, typically one activity in one file

What does an Activity do?do?

- Represents an activity, such as ordering groceries, sending email, or getting directions
- Handles user interactions, such as button clicks, text entry, or login verification
- Can start other activities in the same or other apps
- Has a life cycle—is created, started, runs, is paused, resumed, stopped, and destroyed

Examples of activities



Apps and activities

- Activities are loosely tied together to make up an app
- First activity user sees is typically called "main activity"
- Activities can be organized in parent-child relationships in the Android manifest to aid navigation

Layouts and Activities

- An activity typically has a UI layout
- Layout is usually defined in one or more XML files
- Activity "inflates" layout as part of being created

Implementing Activities

Implement new activities

1. Define layout in XML
2. Define Activity Java class
 - extends AppCompatActivity
3. Connect Activity with Layout
 - Set content view in onCreate()
4. Declare Activity in the Android manifest

1. Define layout in XML

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Let's Shop for Food!" />
</RelativeLayout>
```

Creating an Activity

- ❑ To create an activity, you must create a subclass of **Activity**
- ❑ In your subclass, you need to implement **callback** methods that the system calls when the activity transitions between various states of its lifecycle such as
- ❑ when the activity is being created, stopped, resumed, or destroyed.
 - The two most important callback methods are:
 - **onCreate()**
 - **onPause()**

Methods in Activity

- **onCreate()** You must implement this method.
- The system calls this when creating your activity.
- Within your implementation, you should initialize the essential components of your activity.
- Most importantly, this is where you must call **setContentView()** to define the layout for the activity's user interface.
- **onPause()** The system calls this method as the first indication that the user is leaving your activity (though it does not always mean the activity is being destroyed).
- This is usually where you should commit any changes that should be persisted beyond the current user session .

2. Define Activity Java class

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
    }  
}
```

3. Connect activity with layout

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
setContentView(R.layout.activity_main);  
    }  
}
```

Resource is layout in this XML file

4. Declare activity in Android manifest

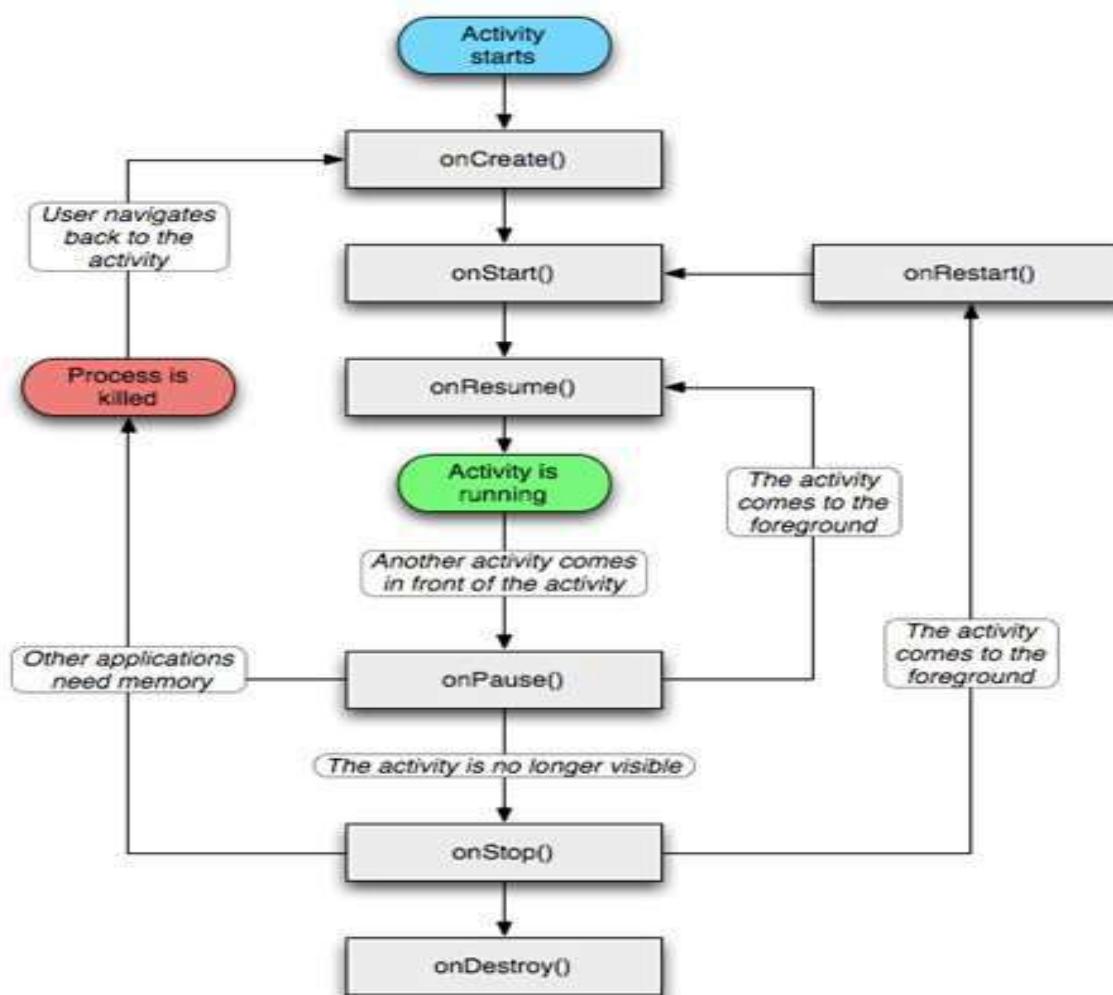
```
<activity  
    android:name=".MainActivity">
```

4. Declare main activity in manifest

Main Activity needs to include intent to start from launcher icon

```
<activity android:name=".MainActivity">  
    <intent-filter>  
        <action android:name="android.intent.action.MAIN" />  
        <category android:name="android.intent.category.LAUNCHER" />  
    </intent-filter>  
</activity>
```

Activity life cycle



Other Methods

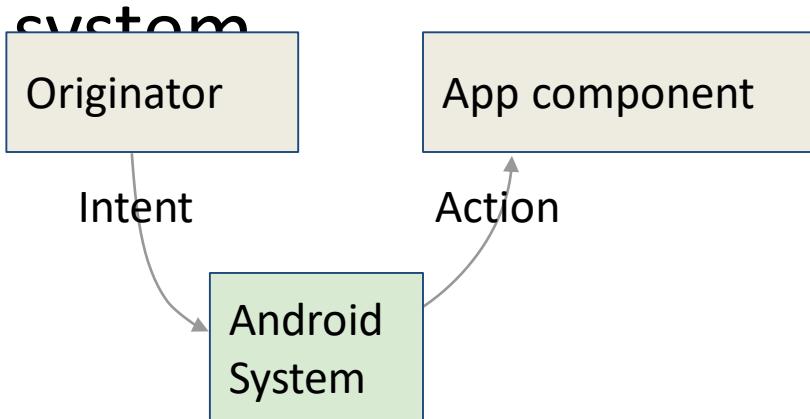
- There are several other lifecycle callback methods :
- **onResume()**
- **onStart()**
- **onStop()**
- **onDestroy()**

Intents

What is an intent?

An intent is a description of an operation to be performed.

An Intent is an object used to request an action from another app component via the Android



What is an Intent?

- Intent is an intention to do something.
 - Intent contains an action carrying some information.
 - Intent is used to communicate between android components.
- To start an activity
- To start a service
- To deliver a broadcast.

What can intents do?

- Start activities
 - A button click starts a new activity for text entry
 - Clicking Share opens an app that allows you to post a photo
- Start services
 - Initiate downloading a file in the background
- Deliver broadcasts
 - The system informs everybody that the phone is now charging

Types of Intents?

- Explicit Intents
- Implicit Intents

Explicit Intents

- Used to launch a specific component like activity or a service.
 - In this case, android system directly forwards this intent to that specific component.
 - It is faster.
 - Always use explicit intents if you know the specific activity or service that performs

Implicit Intent

- Specifies an action that can invoke an app on the device that can perform the action.
 - Useful when your app can not perform the action but other apps do and you let user to pick up the app.
 - Its possible that there may not be any app that handles the implicit intent.

Explicit and implicit intents

Explicit Intent

- Starts a specific activity
 - Request tea with milk delivered by Nikita
 - Main activity starts the ViewShoppingCart activity

Implicit Intent

- Asks system to find an activity that can handle this request
 - Find an open store that sells green tea
 - Clicking Share opens a chooser with a list of apps

How Intents are received?

- Till now we have seen how intents are used to invoke some other components. Now lets explore how these components receive these intents.
 - Receiving Implicit Intents
 - Receiving Explicit Intents
- Explicit Intents are directly delivered to target as intent has the target component class specified in it.

Receiving Implicit Intents

- Your app should advertise what intents it can handle using `<intent-filter>` tag in the manifest file under `<activity>` or `<service>` or `<receiver>` tags.
- You can mention one or more of these three elements under `<intent-filter>`
 - action
 - data
 - category

Starting Activities

Start an Activity with an explicit intent

To start a specific activity, use an explicit intent

1. Create an intent

- `Intent intent = new Intent(this, ActivityName.class);`

2. Use the intent to start the activity

- `startActivity(intent);`

Start an Activity with implicit intent

To ask Android to find an Activity to handle your request, use an implicit intent

1. Create an intent

- o `Intent intent = new Intent(action, uri);`

2. Use the intent to start the activity

- o `startActivity(intent);`

Implicit Intents - Examples

Show a web page

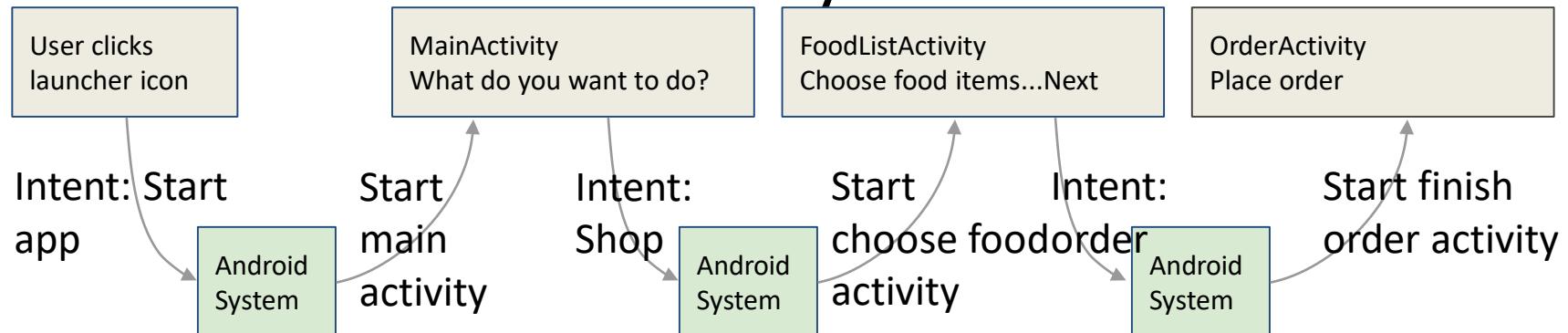
```
Uri uri = Uri.parse("http://www.google.com");  
Intent it = new Intent(Intent.ACTION_VIEW,uri);  
startActivity(it);
```

Dial a phone number

```
Uri uri = Uri.parse("tel:8005551234");  
Intent it = new Intent(Intent.ACTION_DIAL, uri);  
startActivity(it);
```

How Activities Run

- All activities are managed by the Android runtime
- Started by an "intent", a message to the Android runtime to run an activity



Sending and Receiving Data

Two types of sending data with intents

- Data—one piece of information whose data location can be represented by an URI
- Extras—one or more pieces of information as a collection of key-value pairs in a Bundle

Sending and retrieving data

In the first (sending) activity:

1. Create the Intent object
2. Put data or extras into that intent
3. Start the new activity with `startActivity()`

In the second (receiving) activity,:
:

1. Get the intent object the activity was started with
2. Retrieve the data or extras from the Intent object

Putting a URI as intent data

```
// A web page URL  
intent.setData(  
    Uri.parse("http://www.google.com"));  
  
// a Sample file URI  
intent.setData(  
    Uri.fromFile(new  
        File("/sdcard/sample.jpg")));
```

Put information into intent extras

- `putExtra(String name, int value)`
⇒ `intent.putExtra("level", 406);`
- `putExtra(String name, String[] value)`
⇒ `String[] foodList = {"Rice", "Beans", "Fruit"};`
`intent.putExtra("food", foodList);`
- `putExtras(bundle);`
⇒ if lots of data, first create a bundle and pass the bundle.
- See [documentation](#) for all

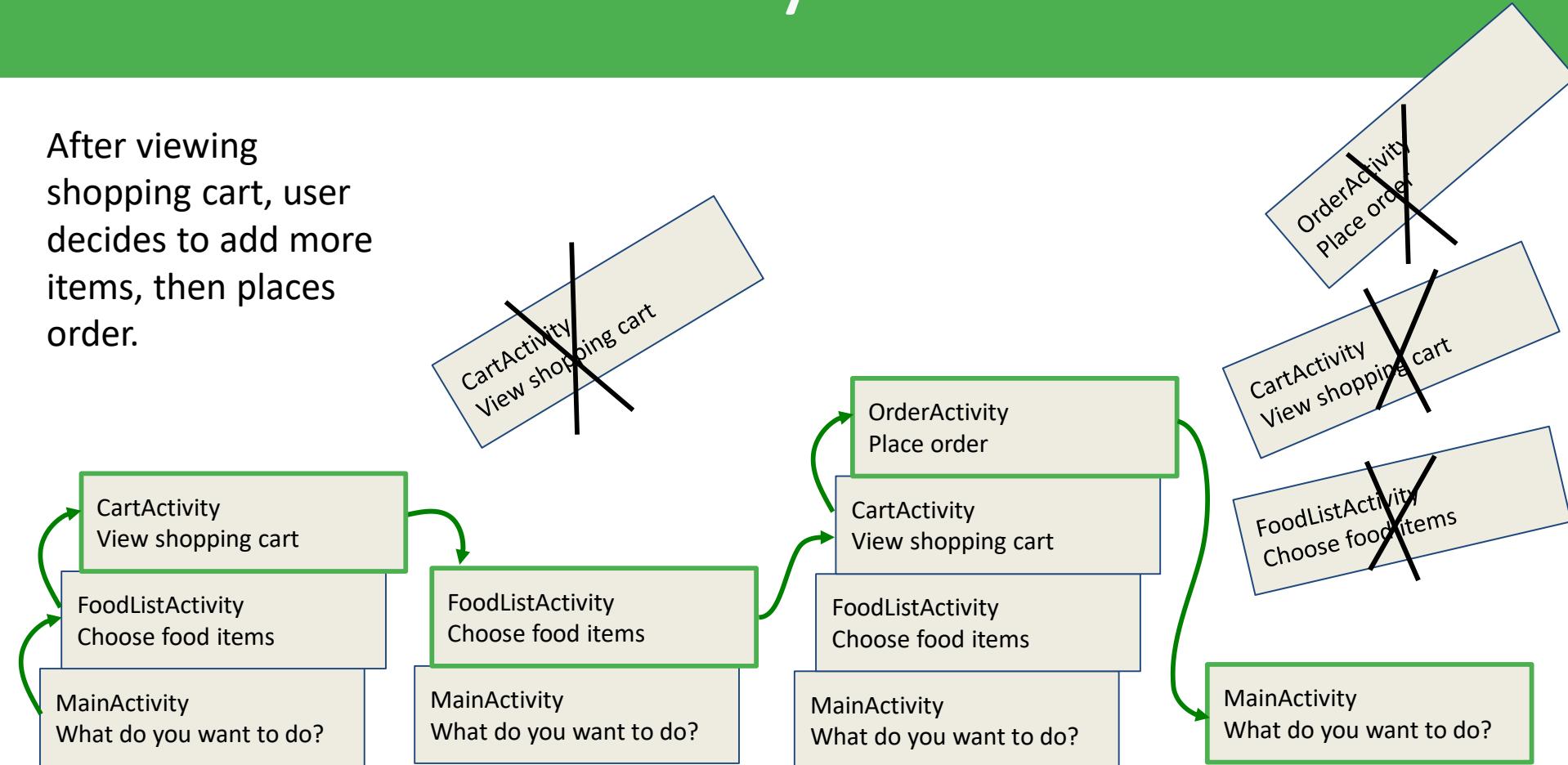
Navigation

Activity stack

- When a new activity is started, the previous activity is stopped and pushed on the activity back stack
- Last-in-first-out-stack—when the current activity ends, or the user presses the Back  button, it is popped from the stack and the previous activity resumes

Activity Stack

After viewing shopping cart, user decides to add more items, then places order.



Two forms of navigation



Temporal or back navigation

- provided by the device's back button
- controlled by the Android system's back stack



Ancestral or up navigation

- provided by the app's action bar
- controlled by defining parent-child relationships between activities in the Android manifest



Back navigation

- Back stack preserves history of recently viewed screens
- Back stack contains all the activities that have been launched by the user in reverse order *for the current task*
- Each task has its own back stack
- Switching between tasks activates that task's back stack
- Launching an activity from the home screen starts a new task
- Navigate between tasks with the overview or recent tasks screen



Up navigation

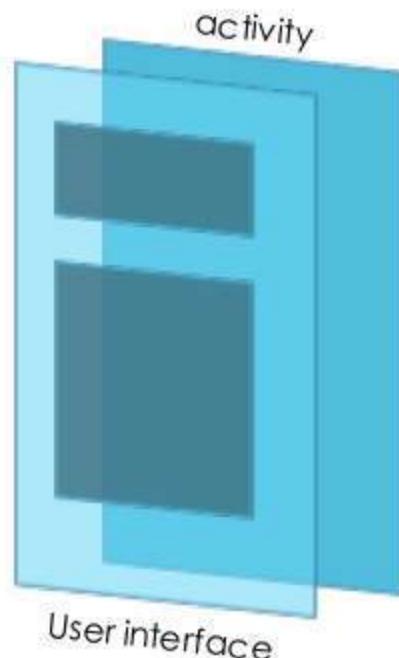
- Goes to parent of current activity
- Define an activity's parent in Android manifest
- Set `parentActivityName`

```
<activity  
    android:name=".ShowDinnerActivity"  
    android:parentActivityName=".MainActivity" >  
</activity>
```

Views and ViewGroups

User interface

- ▣ Each Activity in the app is associated with a user interface (UI)

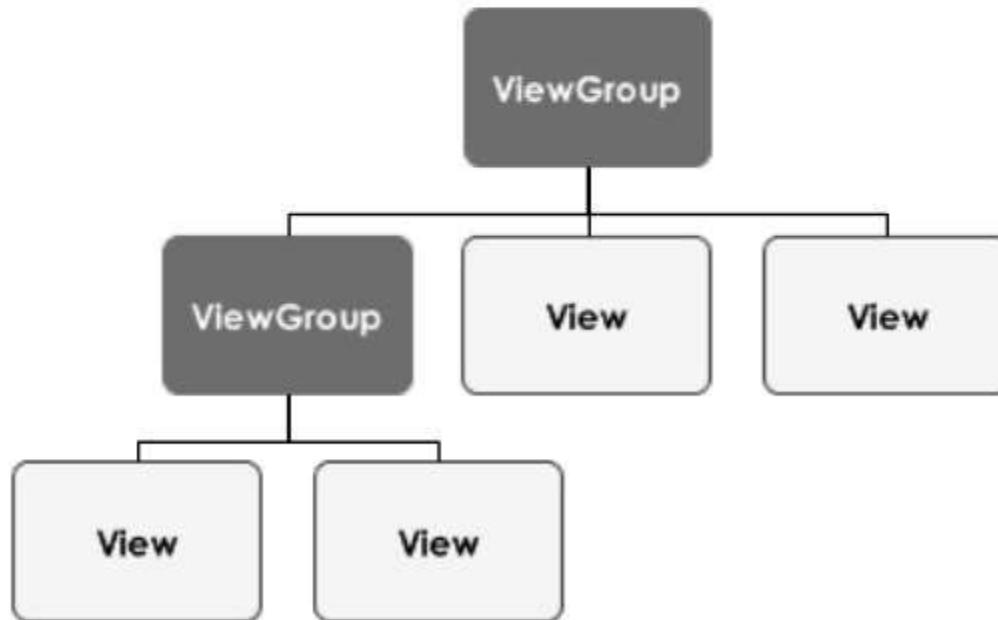


User interface: View and ViewGroup

- User interfaces are built using `View`'s and `ViewGroup`'s:
 - A **View** is a widget that has an appearance on the screen
 - A **ViewGroup** is a special View that contains other Views in order to define the layout of the interface
- In other words, each `ViewGroup` is an **invisible container** that organizes child `Views`, and `Views` are any widget that draws some part of the UI

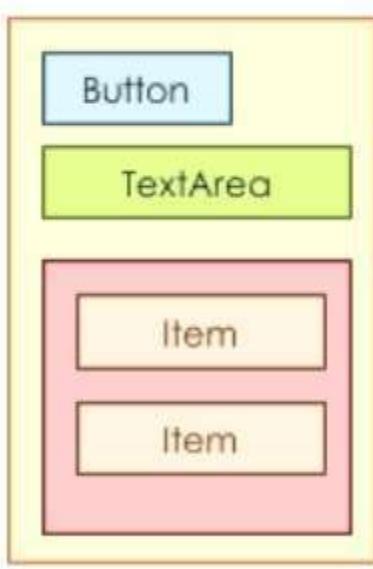
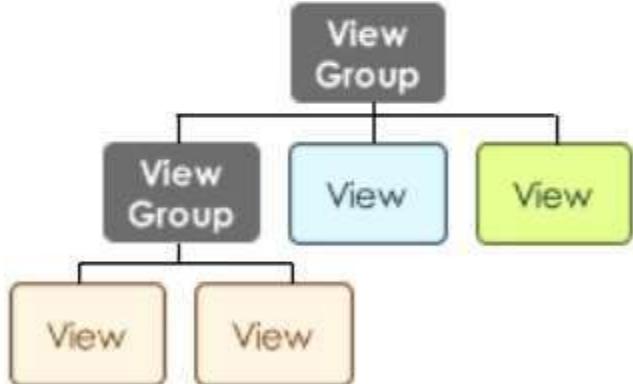
User interface: View and ViewGroup

- The UI for each component of the app is defined using a hierarchy of View and ViewGroup objects



User Interface: View and ViewGroup

- The UI for each component of the app is defined using a hierarchy of View and ViewGroup objects



View types

- Adapter View
- List View
- Picker view

Adapter View

- The **ListView** and **GridView** are subclasses of **AdapterView**
- They can be populated by binding them to an **Adapter**, which retrieves data from an external source and creates a View that represents each data entry.
- An Adapter View can be used to display large sets of data efficiently in form of List or Grid etc, provided to it by an Adapter.
- An Adapter View is capable of displaying millions of items on the User Interface
- While keeping the memory and CPU usage very low and without any noticeable lag

How it works?

- It only renders those View objects which are currently on-screen or are about to some on-screen.
- Hence no matter how big your data set is, the Adapter View will always load only 5 or 6 or maybe 7 items at once, depending upon the display size.
- Hence saving memory.
- It also reuses the already created layout to populate data items as the user scrolls, hence saving the CPU usage.

- Suppose you have a dataset, like a String array with the following contents.
- `String days[] = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};`
- an **Adapter** takes the data from this array and creates a **View** from this data and then, it gives this **View** to an **AdapterView**.
- The AdapterView then displays the data in the way you want.
- You can display it vertically (ListView), or in rows and columns (GridView), or in drop-down menu (Spinners), etc.

List View

- A view which groups several items and display them in vertical scrollable list.
- The list items are automatically inserted to the list using an **Adapter** that pulls content from a source such as an array or database.

List View



Picker view

- Android provides controls for the user to pick a time or pick a date as ready-to-use dialogs
- Each picker provides controls for selecting each part of the time (hour, minute, AM/PM) or date (month, day, year).
- Using these pickers helps ensure that your users can pick a time or date that is valid, formatted correctly, and adjusted to the user's locale.

Set time

17 : 30 AM

8 : 31 PM

Cancel

Set

Set date

Sep 06 2010

Oct 07 2011

Nov 08 2012

Cancel

Set

Menus

Types of Menus

1. App bar with options menu



2. Contextual menu

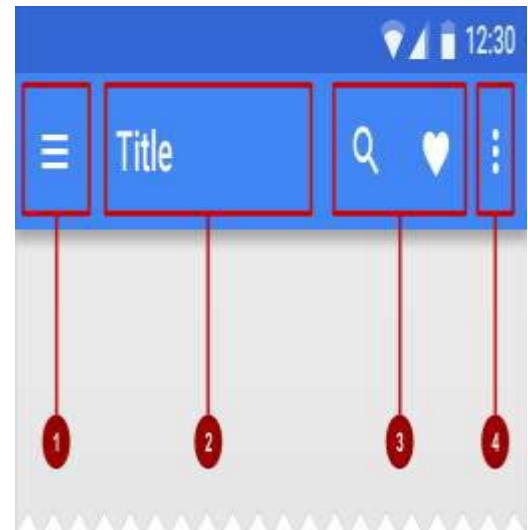
3. Contextual action bar

4. Popup menu

What is the App Bar?

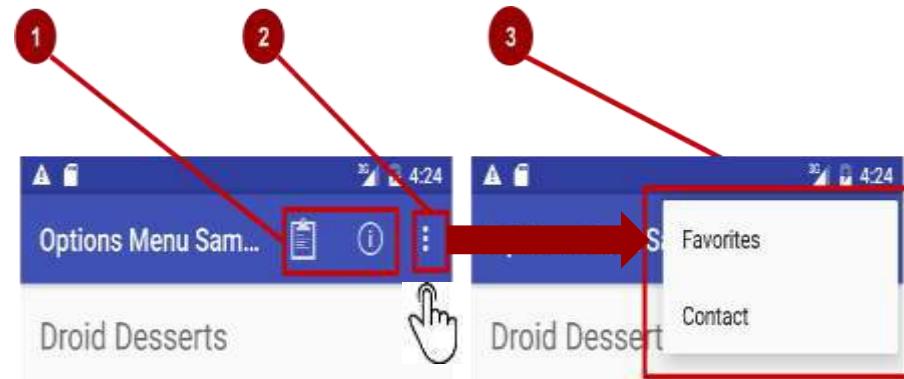
Bar at top of each screen—(usually) the same for all screens

1. Nav icon to open navigation drawer
2. Title of current activity
3. Icons for options menu items
4. Action overflow button for the rest of the options menu



What is the options menu?

- Action icons in the app bar for important items (1)
- Tap the three dots, the "action overflow button" to see the options menu (2)



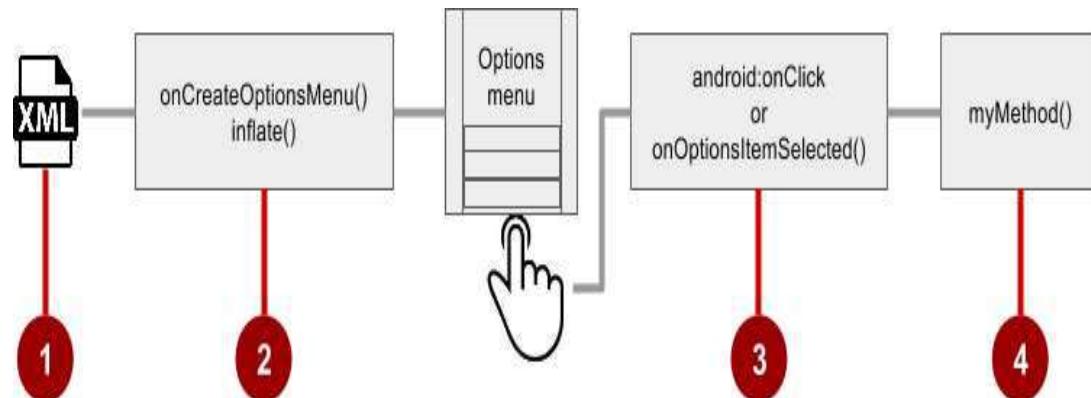
Appears in the right corner of the app bar (3)

- For navigating to other activities and editing app settings

Adding Options Menu

Steps to implement options menu

1. XML menu resource (`menu_main.xml`)
2. `onCreateOptionsMenu()` to inflate the menu
3. `onClick` attribute or `onOptionsItemSelected()`
4. Method to handle item click



Create menu resource

1. Create menu resource directory
2. Create XML menu resource (`menu_main.xml`)
3. Add an entry for each menu item

```
<item android:id="@+id/option_settings"  
      android:title="@string/settings" />  
  
<item android:id="@+id/option_toast"  
      android:title="@string/toast" />
```

Add icons for menu items

1. Right-click **drawable**
2. Choose **New > Image Asset**
3. Choose **Action Bar and Tab Items**
4. Edit the icon name
5. Click clipart image, and click icon
6. Click **Next**, then **Finish**



Add menu item attributes

```
<item android:id="@+id/action_order"
      android:icon="@drawable/ic_toast_dark"
      android:title="@string/toast"
      android:titleCondensed="@string/toast_condensed"
      android:orderInCategory="1"
      app:showAsAction="ifRoom" />
```

Layouts in Android

- **Layouts define the arrangement of views.**
- • **Android layouts are defined mostly in XML.**
- • **Various properties of Layouts can be set**
- **using XML attributes.**

Types Of Layout

- • **Linear Layout**
- • **Relative Layout**
- • **Absolute Layout**
- • **Table Layout**

User interface layout: LinearLayout

- A LinearLayout organizes its children into:
 - Either a single horizontal row
 - Or a single vertical row



To configure a `LinearLayout`, you have five main areas of control besides the container's contents:

- orientation,
- fill model,
- weight,
- gravity, and
- padding

Orientation:

Add the `android:orientation` property to your `LinearLayout` element in your XML layout, setting the value to be `horizontal` for a row or `vertical` for a column.

The orientation can be modified at runtime by invoking
`setOrientation()`

Orientation



Fill-model

All widgets inside a LinearLayout must supply dimensional attributes `android:layout_width` and `android:layout_height` to help address the issue of empty space.

Values used in defining height and width are:

1. Specific a *particular dimension*, such as **125px** to indicate the widget should take up exactly 125 pixels.
2. Provide **wrap_content**, which means the widget should fill up its natural space, unless that is too big, in which case Android can use **word-wrap** as needed to make it fit.
3. Provide **fill_parent**, which means the widget should fill up all available space in its enclosing container, after all other widgets are taken care of.

weight

You set `android:layout_weight` to a value (1, 2, 3, ...) to indicates what proportion of the free space should go to that widget.

Example

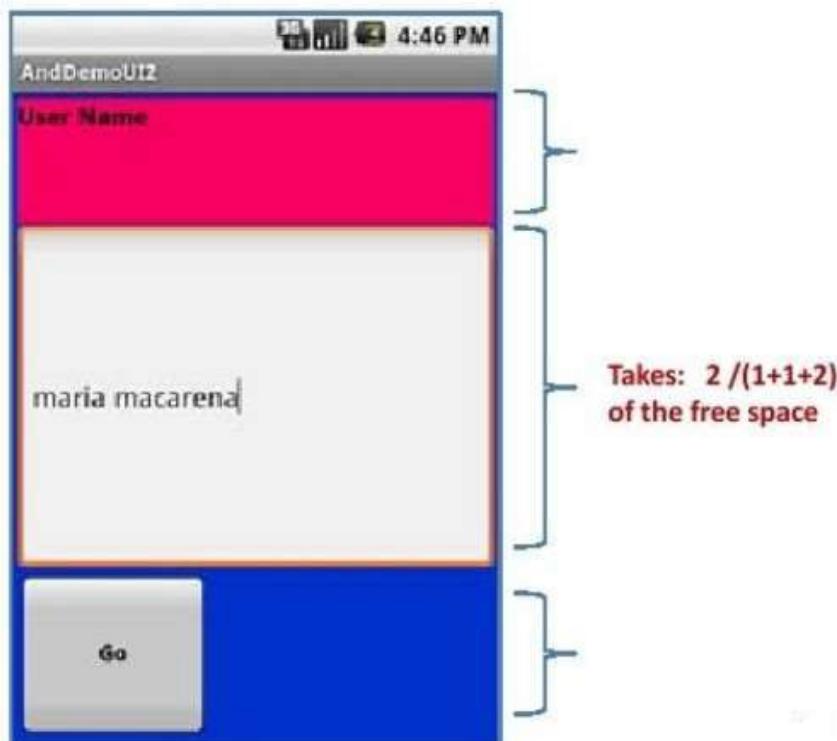
Both the `TextView` and the `Button` widgets have been set as in the previous example.

Both have the additional property

`android:layout_weight="1"`

whereas the `EditText` control has

`android:layout_weight="2"`



Gravity

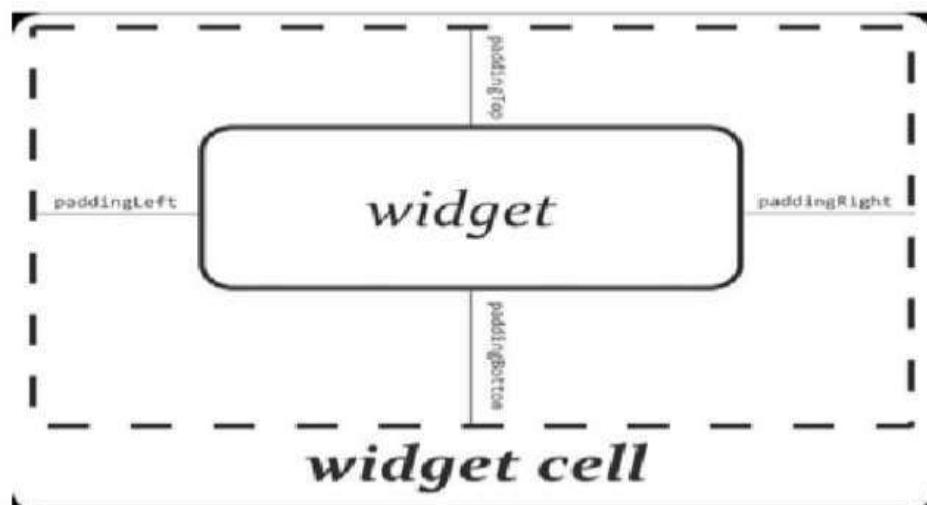
- android:layout_gravity="right"



padding

If you want to increase the whitespace between widgets, you will want to use the `android:padding` property (or by calling `setPadding()` at runtime on the widget's Java object).

The padding specifies how much space there is between the boundaries of the widget's "cell" and the actual widget contents.



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    >

    <TextView android:text="@string/hello_world"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button android:text="Click Me"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Examples



android:orientation="vertical"

android:orientation="horizontal"

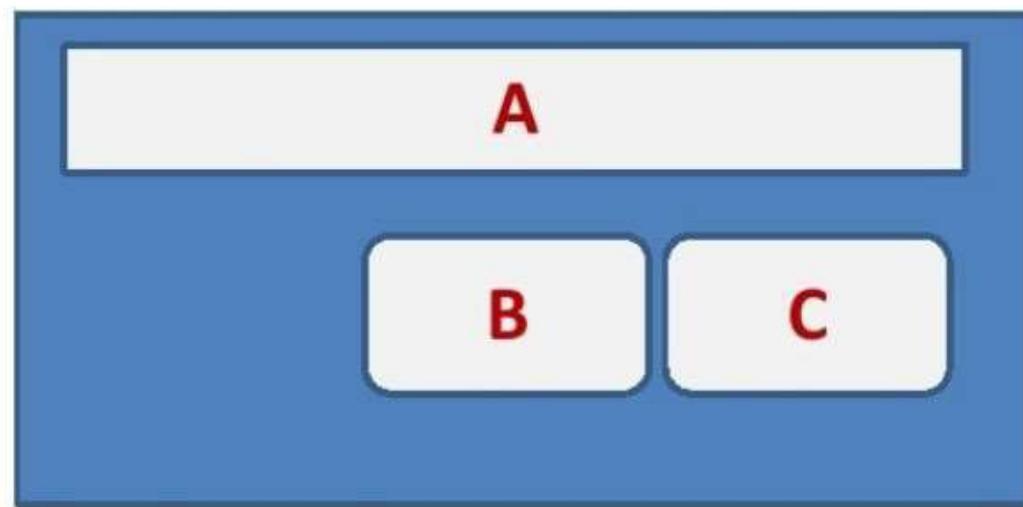
User interface layout: RelativeLayout

- ❑ A RelativeLayout enables you to specify the location of child objects:
 - ❑ either relative to each other
 - ❑ or relative to the parent



RelativeLayout

RelativeLayout places widgets based on their relationship to other widgets in the container and the parent container.



Example:

A is by the parent's top

C is below A, to its right

B is below A, to the left of C



Relative Layout attributes

- **android:layout_centerHorizontal** the widget should be positioned horizontally at the center of the container
- **android:layout_centerVertical** the widget should be positioned vertically at the center of the container
- **android:layout_centerInParent** the widget should be positioned both horizontally and vertically at the center of the container
- **android:layout_above** indicates that the widget should be placed above the widget referenced in the property
- **android:layout_below** indicates that the widget should be placed below the widget referenced in the property
- **android:layout_toLeftOf** indicates that the widget should be placed to the left of the widget referenced in the property
- **android:layout_toRightOf** indicates that the widget should be placed to the right of the widget referenced in the property

Relative Layout

Put identifiers (`android:id` attributes) on *all elements* that you will need to address. Syntax is: `@+id/...` (for instance an EditText box could be XML called: `android:id="@+id/ediUserName"`)

Reference other widgets using the same identifier value (`@+id/...`) already given to a widget. For instance a control below the EditText box could say:
`android:layout_below="@+id/ediUserName"`

Relative Layout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/myRelativeLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff000099"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_alignParentLeft="true"
    android:padding="20px">

    <EditText
        android:id="@+id/ediUserName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ffff0066"
        android:text="User Name"
        android:textStyle="bold"
        android:textColor="#ff000000"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true">
    </EditText>

    <Button
        android:id="@+id/btnGo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/ediUserName"
        android:layout_alignRight="@+id/ediUserName"
        android:text="Go"
        android:textStyle="bold">
    </Button>

    <Button
        android:id="@+id btnCancel"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@+id/btnGo"
        android:layout_below="@+id/ediUserName"
        android:text="Cancel"
        android:textStyle="bold">
    </Button>
</RelativeLayout>
```



TableLayout

1. Android's **TableLayout** allows you to position your widgets in a grid made of identifiable *rows* and *columns*.
2. Columns might *shrink* or *stretch* to accommodate their contents.
3. TableLayout works in conjunction with **TableRow**.
4. TableLayout controls the overall behavior of the container, with the widgets themselves positioned into one or more *TableRow* containers, one per row in the grid.

Table Layout

```
<TableRow>
<TextView
    android:text="URL:" />
<EditText android:id="@+id/ediUrl"
    android:layout_span="3"/>
</TableRow>
<View
    android:layout_height="3px"
    android:background="#0000FF" />
<TableRow>
<Button android:id="@+id/cancel"
    android:layout_column="2"
    android:text="Cancel" />
<Button android:id="@+id/ok"
    android:text="OK" />
</TableRow>
```

Stretch up to column 3

Skip column: 1

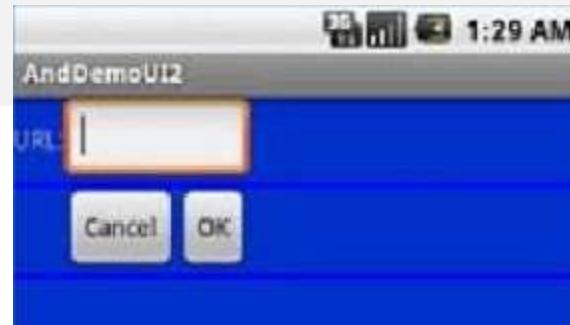


Table Layout

If your content is narrower than the available space, you can use the *TableLayout* property:

```
android:stretchColumns = "..."
```

Its value should be a single column number (0-based) or a comma-delimited list of column numbers. Those columns will be stretched to take up any available space yet on the row.

```
...
<TableLayout
    android:id="@+id/myTableLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff0033cc"
    android:orientation="vertical"
    android:stretchColumns ="2,3,4"
    xmlns:android="http://schemas.android.com/apk/res/android"
>
    ...

```



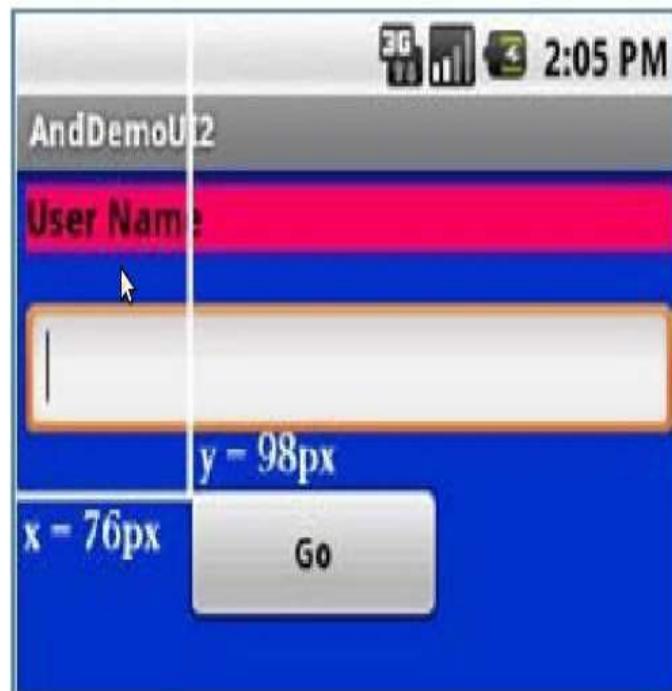
TableLayout Example

```
<TableLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content">  
  
    <TableRow  
        android:id="@+id/row1"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:padding="5dip" >  
  
        <TextView android:id="@+id/text1" android:text="Column 1" />  
  
        <Button android:id="@+id/btn1" android:text="Column 2" />  
  
        <Button android:id="@+id/btn2" android:text="Column 2" />  
  
    </TableRow>  
    <TableRow  
        android:id="@+id/row2"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:padding="5dip" >  
  
        <TextView android:id="@+id/text2" android:text="Column 1"/>  
  
        <Button android:id="@+id/btn3" android:text="Column 2"  
            android:layout_span="2"/>  
  
        <Button android:id="@+id/btn4" android:text="Column 2" />  
    </TableRow>  
  
</TableLayout>
```



Absolute Layout

- A layout that lets you specify exact locations (x/y coordinates) of its children.
- Absolute layouts are *less flexible* and harder to maintain than other types of layouts without absolute positioning.



Absolute Layout

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/myLinearLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ff0033cc"
    android:padding="4px"
    xmlns:android="http://schemas.android.com
    /apk/res/android"
>

<TextView
    android:id="@+id/labelUserName"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#ffff0066"
    android:text="User Name"
    android:textSize="16sp"
    android:textStyle="bold"
    android:textColor="#ff000000"
    android:layout_x="0px"
    android:layout_y="-1px"
>
    </TextView>
    <EditText
        android:id="@+id/ediName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:layout_x="0px"
        android:layout_y="38px"
    >
    </EditText>
    <Button
        android:id="@+id/btnGo"
        android:layout_width="125px"
        android:layout_height="wrap_content"
        android:text="Go"
        android:textStyle="bold"
        android:layout_x="76px"
        android:layout_y="98px"
    >
    </Button>
</AbsoluteLayout>
```

Button
location

History



- SQLite is an open source embedded database. The original implementation was designed by D. Richard Hipp.

Why Sqlite Only????



- In android we will use Sqlite database only . Because it is in built DB in Android SDK more over it is lite weighted relation DBsuitable for Mobile Devices.
- We need not to load any drivers and we need not to install the Sqlite separately.
- The queries also simple to understand and easy to implement.



Embedded RDBMS (Relational Database Management System)

- ACID Compliant (Atomicity, Consistency, Isolation, Durability)
- Size – about 257 Kbytes
- Not a client/server architecture
- Accessed via function calls from the application
- Writing (insert, update, delete) locks the database, queries can be done in parallel

Feature of SQLite



- **Application file format** – Transactions guarantee ACID [Atomicity, Consistency , Isolation, Durability] even after system crashes and power failures.
triggers provide undo/redo feature.
- **Temporary data analysis** – Command line client, import CSV files and use sql to analyze & generate reports .
- **Embedded devices** – Applicable to small, reliable and portable like mobiles.
- **Portable** - uses only ANSI-standard C and VFS, file format is cross platform (little vs. big endian, 32 vs. 64 bit)

Feature of SQLite



- **Reliable** – has 100% test coverage, open source code and bug database, transactions are ACID even if power fails.
- **Small** – 300 kb library, runs in 16kb stack and 100kb heap.
- **Single Database File** – An SQLite database is a single ordinary disk file that can be located anywhere in the directory hierarchy.
- **Readable source code** – The source code to SQLite is designed to be readable and accessible to the average programmer.



Unique Features.

- No configuration. Just drop in the C library and go.
- No server process to administer or user accounts to manage.
- Easy to backup and transmit database (just copy the file)
- Dynamic typing for column values, variable lengths for column records
- Query can reference multiple database files
- A few non-standard SQL extensions (mostly for conflict resolution)

Disadvantages



- **High concurrency** – reader/writer locks on the entire file.
- **Huge datasets** – DBfile can't exceed file system limit or 2TB.
- **Access control** – we don't have any user interface to operate Sqlite database objects as in MYSQL/ SQLServer /Oracle. All the objects are virtual. However there are few third party UI are available in the market.

Examples To Do



Creating Sqlite Object

Step:1 Importing package

“android.database.sqlite.SQLiteDatabase”.

Step:2 Creating object

SQLiteDatabase object name=null;

Examples To Do



Let us do some coding

Creating Database Name

```
EX:mydb=openOrCreateDatabase("DatabaseName5",  
    MODE_PRIVATE,null);
```

//mydb is sqlite object name.
//DatabaseName5 is nothing but database name
//MODE_PRIVATE is permissions of a table accessing

Examples To Do



Let us do some coding

Creating Table

```
mydb.execSQL("CREATE TABLE IF NOT EXISTS"  
+TableName+" (ColumnName  DataType);");
```

Examples To Do



Let us do some coding

DDL(Create ,Alter ,Drop)

Create:

```
mydb.execSQL("CREATE TABLE IF NOT EXISTS"  
+TableName+" (ColumnName DataType);");
```

Alter:

```
ALTER TABLE TableName RENAME TO new-table-name
```

Drop:

```
DROP TABLE TableName
```

([View Source](#))

Examples To Do



Let us do some coding

DML(SELECT , INSERT ,
DELETE)

Select:

```
Cursor c=mydb.rawQuery("SELECT * FROM"+TableName+  
"where Name='"+city+"'",null);
```

Insert:

```
mydb.execSQL("INSERT INTO "+TableName+" (Name, Area) "+  
"VALUES ('RedFort','40.8 acres');");
```

Delete:

```
mydb.execSQL("Delete"+TableName);
```

([View Source](#))

SQLiteOpenHelper



- SQLiteOpenHelper is a class to manage database creation and version management.
- This class takes care of opening the database if it exists, creating it if it does not, and upgrading it as necessary.
- This is for creating db “**onCreate(SQLiteDatabase)**”.
- when the database needs to be upgraded
“**onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion)**”.
- when the database has been opened
“**onOpen (SQLiteDatabase db)**”.

android.database.sqlite



- Contains the SQLite database management classes that an application would use to manage its own private database.

android.database.sqlite - Classes



- **SQLiteCloseable** - An object created from a SQLiteDatabase that can be closed.
- **SQLiteCursor** - A Cursor implementation that exposes results from a query on a SQLiteDatabase.
- SQLiteDatabase** - Exposes methods to manage a SQLite database.
- **SQLiteOpenHelper** - A helper class to manage database creation and version management.
- **SQLiteProgram** - A base class for compiled SQLite programs.
- **SQLiteQuery** - A SQLite program that represents a query that reads the resulting rows into a CursorWindow.
- **SQLiteQueryBuilder** - a convenience class that helps build SQL queries to be sent to SQLiteDatabase objects.
- **SQLiteStatement** - A pre-compiled statement against a SQLiteDatabase that can be reused.

Storage classes



- **NULL** – null value
- **INTEGER** - signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value
- **REAL** - a floating point value, 8-byte IEEE floating point number.
- **TEXT** - text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).
- **BLOB** -The value is a blob of data, stored exactly as it was input

openOrCreateDatabase()



- This method will open an existing database or create one in the application data area

```
import android.database.sqlite.SQLiteDatabase;
SQLiteDatabase myDatabase;
myDatabase = openOrCreateDatabase
("my_sqlite_database.db" ,
SQLiteDatabase.CREATE_IF_NECESSARY , null);
```

Creating Tables



- Create a static string containing the SQLite CREATE statement, use the execSQL() method to execute it.

```
String createAuthor = "CREATE TABLE authors (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    fname TEXT,  
    lname TEXT);  
myDatabase.execSQL(createAuthor);
```

insert()



```
long insert(String table, String nullColumnHack,  
ContentValues values)
```

```
import android.content.ContentValues;  
ContentValues values = new ContentValues( );  
values.put("firstname" , "J.K.");  
values.put("lastname" , "Rowling");  
long newAuthorID = myDatabase.insert("tbl_authors" , "" ,  
values);
```

update()



```
int update(String table, ContentValues values, String  
whereClause, String[ ]  
whereArgs)
```

```
public void updateBookTitle(Integer bookId, String newTitle)  
{  
    ContentValues values = new ContentValues();  
    values.put("title" , newTitle);  
    myDatabase.update("tbl_books" , values ,  
    "id=? " , new String[ ] {bookId.toString() } );  
}
```

delete()



```
int delete(String table, String whereClause, String[]  
whereArgs)
```

```
public void deleteBook(Integer bookId) {  
    myDatabase.delete("tbl_books" , "id=?" ,  
    new String[ ] { bookId.toString( ) } ) ;
```



Map

- Google owns Android, so it is obvious that the map you're going to use in Android app must be Google Map.
- However to use Google Map in your app is not as easy as it should be.

Instruction how to make Map app:



1. Goto window menu -> Android SDK Manager
-> Download Google API (2.3.3 API 10)

Android SDK Manager

Packages Tools

SDK Path: C:\Program Files (x86)\Android\android-sdk\

Packages

Name	Version	Status
XOOM by Motorola Mobility, Inc.	10	Not installed
Android 2.3.3 (API10)	10	Not installed
SDK Platform	10	Not installed
Samples for SDK	10	Not installed
Dual Screen APIs by KYOCERA Corporation	10	Not installed
Real3D by LGE	10	Not installed
ATRIX2 by Motorola Mobility, Inc.	10	Not installed
Droid4 by Motorola Mobility, Inc.	10	Not installed
DroidX2 by Motorola Mobility, Inc.	10	Not installed
MT917 by Motorola Mobility, Inc.	10	Not installed
XT882 by Motorola Mobility, Inc.	10	Not installed
XT928 by Motorola Mobility, Inc.	10	Not installed
defy+ by Motorola Mobility, Inc.	10	Not installed
MT870 by Motorola Mobility, Inc.	10	Not installed
PHOTON by Motorola Mobility, Inc.	10	Not installed
FDK 1.2 by Sony Ericsson Mobile Communications	10	Not installed
Google APIs by Google Inc.	10	Not installed

Show: Updates/New Installed Obsolete Select [New or Updates](#)

Sort by: API level Repository [Deselect All](#)

Done loading packages.

Choose Packages to Install

Packages

Google APIs by Google Inc., Android API1...

Package Description & License

Package Description
Android + Google APIs
Revision 2
Requires SDK Platform Android API14

This update will replace revision 1 with revision 2.

Archive Description
Archive for any OS
Size: 101.6 MiB
SHA1: f8eb4d96ad0492b4c0db2d7e4f1a1a3836664d39

Accept Reject Accept All

Install Cancel



2. Create AVD Targeting Google API:

For this map app, you need to create an AVD targeting Google API. In this case, please choose API level 10 for 2.3.3.

Edit Android Virtual Device (AVD)

Name: GoogleMap

Target: Google APIs (Google Inc.) - API Level 10

CPU/ABI: Android 1.5 - API Level 3
Android 1.6 - API Level 4
Android 2.1 - API Level 7
Android 2.2 - API Level 8
Android 2.3.3 - API Level 10
Google APIs (Google Inc.) - API Level 10

SD Card: Google APIs (Google Inc.) - API Level 10

Snapshot: Google APIs (Google Inc.) - API Level 11
Android 3.0 - API Level 11
Android 3.1 - API Level 12
Android 3.2 - API Level 13
Android 4.0 - API Level 14

Skin: Built-in: HVGA

Resolution: Resolution: x

Hardware:

Property	Value	New...
Abstracted LCD density	160	Delete
Max VM application heap size	24	
Device ram size	256	

Override the existing AVD with the same name

Edit AVD Cancel



3. Android Manifest: Add Google Maps Library

```
<application ...>
```

```
    <uses-library android:name="com.google.android.maps" />
```

```
    <activity...>
```

```
    ...
```

```
    </application>
```

4. Android Manifest: Add Internet Permission

```
<manifest ...>
```

```
    <uses-sdk android:minSdkVersion="10" />
```

```
    <uses-permission
```

```
        android:name="android.permission.INTERNET"/>
```

```
    ...
```

```
    ...
```

```
</manifest>
```



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android= "http://schemas.android.com/apk/res/android" 
    package="com.kosalab"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <uses-library android:name="com.google.android.maps" />
        <activity
            android:label="@string/app_name"
            android:name=".GoogleMapActivity" >
            <intent-filter >
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

5. main.xml: replace with this codes



```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.maps.MapView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/mymap"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:clickable="true"
    android:apiKey="Enter Your API Key"
/>
```

6. Getting API Key

1. Goto a website, <http://code.google.com/android/add-ons/google-apis>

2. Click on Maps API Key Signup

The screenshot shows a web browser window with the following details:

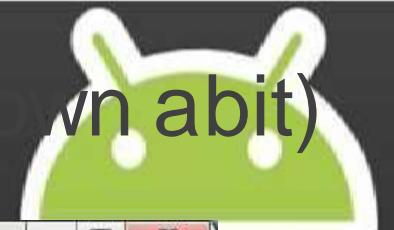
- Title Bar:** Google APIs Add-On - Google Code
- Address Bar:** code.google.com/android/add-ons/google-apis/
- Toolbar:** Back, Forward, Stop, Home, Refresh, Bookmarks, Share FB, Files - SkyDrive, My Delicious, Android References, Search, Other bookmarks.
- Search Bar:** e.g. "adwords" or "open source"
- Page Content:**
 - Header:** Google Projects for Android: Google APIs
 - Left Sidebar (Getting Started):**
 - What is Google APIs Add-on?
 - [Installing the Add-on](#)
 - Left Sidebar (Maps):**
 - [Overview](#)
 - [Obtaining a Maps API Key](#)
 - [**Maps API Key Signup**](#) (This link is circled in red)
 - [API Reference](#)
 - Main Content:**

Google APIs Add-On

Google APIs Add-On is an extension to the Android SDK development environment that lets you develop applications for devices that include Google's set of custom applications, libraries, and services. A central feature of the add-on is the Maps external library, which lets you add powerful mapping capabilities to your Android application.

The add-on also provides a compatible Android system image that runs in the Android Emulator, which lets you debug, profile, and test your application before publishing it to users. The system image includes the the Maps library and other custom system components that your applications may need, to access Google services (such as [Android C2DM](#)). The add-on does not include any custom Google applications. When you are ready to publish your application, you can deploy it to any Android-powered device that runs a compatible version of the

6.3. Enter your MD5 fingerprint. (Scroll down abit)



Maps API Key Signup - Goo x

code.google.com/android/add-ons/google-apis/maps-api-signup.html

Bookmark Share FB Files - SkyDrive My Delicious Android References Other bookmarks

(Google). This legal agreement is referred to as the "Terms."
1.2. Unless otherwise agreed in writing with Google, the Terms will include the following: 1) the terms and conditions set forth in this document (the "Maps APIs Terms"); 2) the Legal Notices (http://www.google.com/intl/en-us/help/legalnotices_maps.html); and 3) the Privacy Policy (<http://www.google.com/privacy.html>). Before you use the Maps APIs, you should read each of the documents comprising the Terms, and print or save a local copy for your records.
1.3. If you use the Maps APIs in conjunction with any other Google products or

I have read and agree with the terms and conditions ([printable version](#))

My certificate's MD5 fingerprint:

Generate API Key

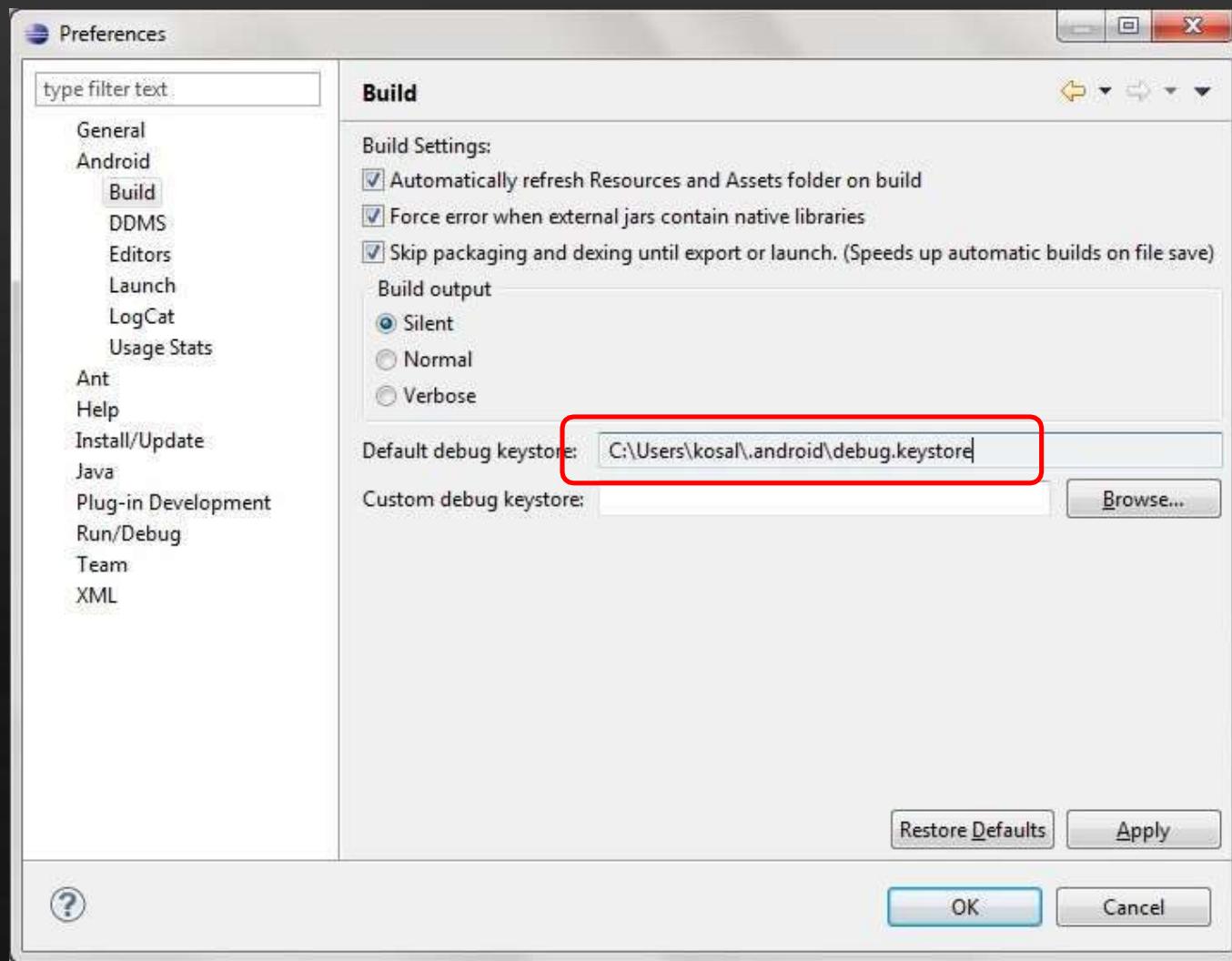
©2012 Google - [Code Home](#) - [Site Terms of Service](#) - [Privacy Policy](#) - [Site Directory](#)

Google Code offered in: [English](#) - [Español](#) - [日本語](#) - [한국어](#) - [Português](#) - [Русский](#) - [中文\(简体\)](#) - [中文\(繁體\)](#)

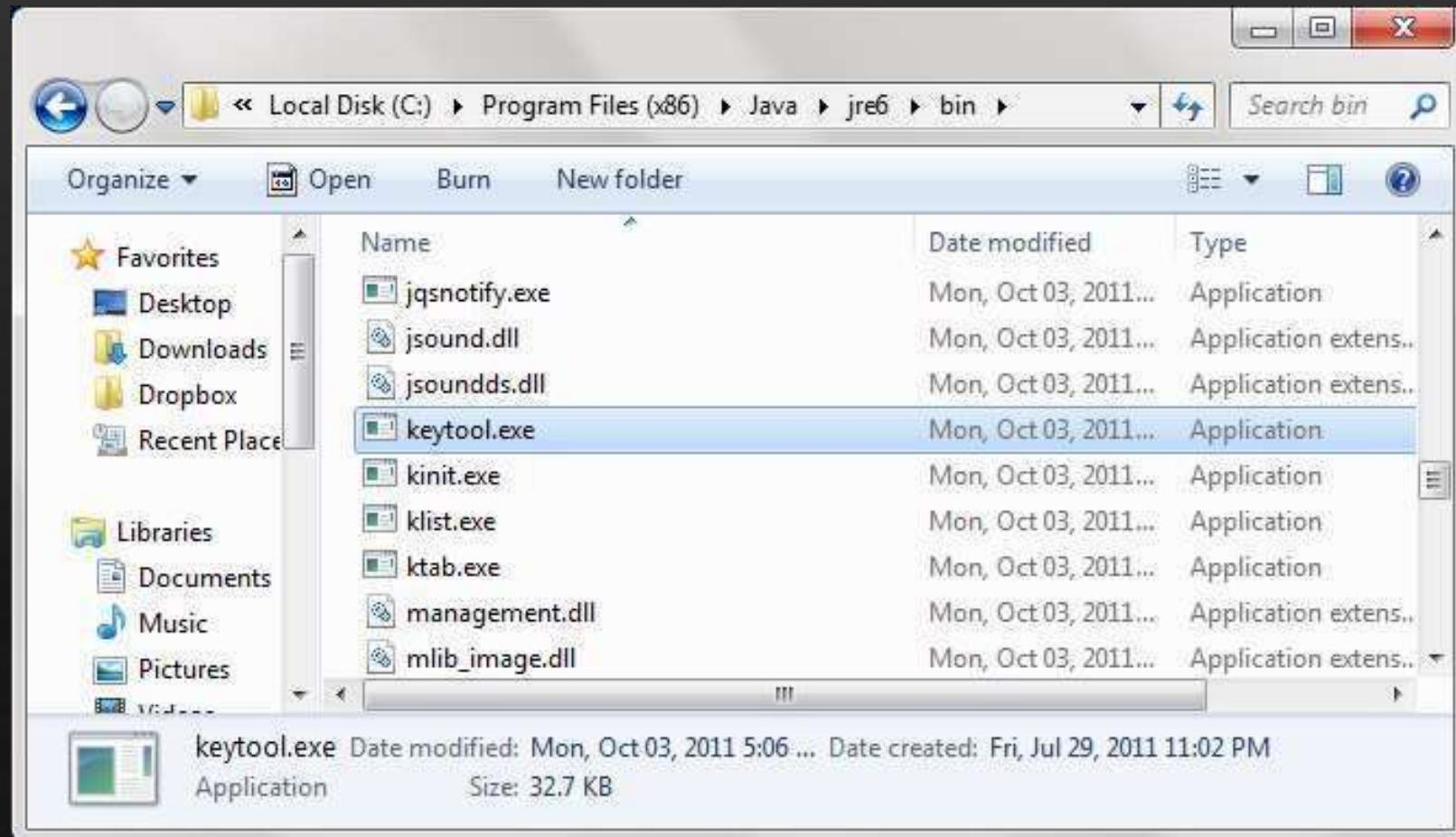


6.4. To get MD5 Fingerprint

6.4.1 In Eclipse, go to Window -> Preferences -> Android -> Build -> Default debug keystore -> Copy the path



6.4.2. In Windows Explorer, go to Java installed folder, e.g. c:\Program Files (x86)\Java -> jre6 -> bin -> keystore.exe



- 6.4.3. After you found out keystore.exe, open cmd, and then go to folder where keystore located.



```
Administrator: C:\Windows\system32\cmd.exe
C:\Program Files (x86)\Java\jre6\bin>_
```



• 6.4.4. Type a command line



Verbose

Default debug keystore:

Custom debug keystore:

```
keytool -list -alias androiddebugkey -keystore  
"C:\Users\kosal\.android\debug.keystore" -storepass  
android -keypass android
```

```
Administrator: C:\Windows\system32\cmd.exe  
  
C:\Program Files (x86)\Java\jre6\bin>keytool -list -alias androiddebugkey -keystore "C:\Users\kosal\.android\debug.keystore" -storepass android -keypass android  
androiddebugkey, Jul 29, 2011, PrivateKeyEntry,  
Certificate fingerprint (MD5): [REDACTED] C9:F2:63: [REDACTED]:4C:C4  
C:\Program Files (x86)\Java\jre6\bin>
```

-> Copy the PrivateKey

- 6.4.5. Paste the PrivateKey into MD5 Fingerprint in the website -> Click "Generate API Key"

The screenshot shows a web browser window with the title "Maps API Key Signup - Google". The address bar contains the URL "code.google.com/android/add-ons/google-apis/maps-api-sign". The page content includes a notice about legal notices and privacy policy, followed by a checkbox for accepting terms and conditions, a text input for the MD5 fingerprint, and a "Generate API Key" button.

Notices (http://www.google.com/intl/en-us/help/legalnotices_maps.html); and 3) the Privacy Policy (<http://www.google.com/privacy.html>). Before you use the Maps API, you must accept the Terms of Service ([View Terms of Service](#)) and the Privacy Policy ([View Privacy Policy](#)).

I have read and agree with the terms and conditions ([printable version](#))

My certificate's MD5 fingerprint:
[REDACTED]C9:F2:63:[REDACTED]4C:C4

[Generate API Key](#)

6.4.6. Finally, you will get your API key:



Android Maps API - Thank' x

www.google.com/glm/mmap/a/api?fp=90%3AC4%3A31%3A3C

Bookmark Share FB Files - SkyDrive My Delicious Android References Other bookmarks

Here is an example xml layout to get you started on your way to mapping glory:

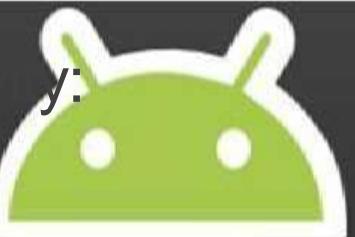
```
<com.google.android.maps.MapView  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:apiKey="[REDACTED]M3qn-GzoA"  
/>
```

Check out the [API documentation](#) for more information..

7. Enter the API key into main.xml



```
<?xml version="1.0" encoding="utf-8"?>  
<com.google.android.maps.MapView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
        android:id="@+id/mymap"  
        android:layout_width="fill_parent"  
        android:layout_height="fill_parent"  
        android:clickable="true"  
        android:apiKey="eNtEr_y0uR_Ap1_kEy_HeRe"  
    />
```



```
Our Java, You must extends MapActivity instead of Activity:
```

```
package com.kosalab;

import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;

import android.os.Bundle;

public class GoogleMapActivity extends MapActivity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        MapView mapView = (MapView) findViewById(R.id.mymap);
        mapView.setBuiltInZoomControls(true);
    }

    @Override
    protected boolean isRouteDisplayed() {
        // TODOAuto-generated method stub
        return false;
    }
}
```



Run the App

