Android Mobile OS

Pradipta De
pradipta.de@sunykorea.ac.kr
Today’s Topic

- Android Basics
- Android Architecture
What is Android?

- Android is a software stack for mobile devices that includes an OS, middleware and key applications.
- It is developed and maintained by the Android Open Source Project (AOSP).
- Android was acquired by Google in 2005, and then modified, and finally Android v1.0 announced in Sep, 2008.
Key parts: **Android compatible Linux kernel**, and **Android Platform**

The Android Platform is essentially a custom Linux distribution containing the userspace packages that make up what is typically called “Android.”
Android is built on top of Linux 2.6 kernel. Android does not use X-windows setup, GNU libraries → cannot easily port standard Linux applications to Android. Android adds the Java abstraction layer between applications and the kernel.
• Application Framework is the API for apps
  – Activity Manager: governs the app lifecycle
  – Content Providers: enable data sharing across apps (access/share data from/with other apps)
  – Resource Manager: manages non-code resources (like graphics)
  – Notification manager: enables apps to display custom alerts

• Android built-in apps, and other user applications, use the same API
Android Native Libraries

• Bionic C library: libc library optimized for embedded platforms
• Surface Manager: compose window manager
• 2D, 3D graphics: support for software simulation
• Media Framework: supports major audio video codecs
• SQLite: database
• Webkit: library for fast HTML rendering
Dalvik Virtual Machine

- Android apps run in a Process VM – called Dalvik Virtual Machine (DVM)
  - replaced by Android Runtime (ART) in latest release Android Lollipop

- DVM uses different byte-code from Java
  - A tool, dx, converts Java class files into Dalvik executables

- DVM features:
  - Memory efficient with low memory footprint for executable
  - Supports a Just-in-Time compiler – not interpreted like Java
  - Register based VM, compared to stack-based VM in Java
New Android Runtime (ART)

The life of an APK
Android Startup Process

- Loads the bootloader from a pre-defined address → bootloader initializes basic hardware, and loads the kernel
  - Kernel can be loaded from a SD card
- Kernel calls the init process
  - Init.rc and init.device.rc is accessed
- Starts another process called zygote
  - Zygote loads core Java classes and waits on a socket for requests to spawn new processes
  - Zygote is the mother of all processes in Android
Startup process in detail

- **Kernel**
  - Init. env. to run C code
  - Init kernel subsystems
  - Init all drivers
  - Mount root FS
  - Start "init" process

- **Bootloader**
  - Initialize RAM
  - Put basic HW in quiescent state
  - Load kernel and RAM disk
  - Jump to kernel

- **Launcher**
  - Init itself
  - Register onClick() handlers

- **Activity Manager**
  - Init itself
  - Send intent.CATEGORY_HOME

- **System Server**
  - For each service:
    - Init service
    - Reg. w/ Service Manager
      - Incl. start Activity Manager

- **Zygote**
  - Register Zygote socket
  - Preload all Java classes
  - Preload resources
  - **Start System Server**
    - Open socket
    - Listen for connections

- **CPU**
  - startActivity()
  - startActivity()

- **Native daemons**
  - servicemanager
  - void
  - netd
  - debuggerd
  - rild
  - app_process-X.Zygote
    - mediaserver
    - bootanimation
    - bluetoothd
    - dbus-daemon
    - installd
    - keystore
    - adbd

- **Android Runtime**
  - Start a Dalvik VM
  - Call Zygote's main()
Important facts

• Android is designed as a single user OS
• Android does not support any OS swap space
• When memory is used up by the apps, some apps are killed to free memory for new apps (uses Linux OOM killer)
• Each app is assigned a unique process id
  – Linux style process management
• All processes are spawned by zygote using fork system call
  – The process address space is shared
  – Since the core Java libraries are read-only, hence due to COW, these libraries are shared across all apps
Putting It Together

• Basic view of Android Internals
  – Android stack and its components
  – Use of Linux and its features