SisInf Lab

Il sistema operativo ANDROID

(libera rielaborazione da

Wikipedia

Android 4 di Massimo Carli

Many others)

A cura di

Giacomo Piscitelli

Bari dicembre 2014

ANDROID

Sommario

1. Generalità

Negli ultimi anni, Android si è affermata come la principale piattaforma per lo sviluppo di applicazioni mobili, dividendosi quasi tutto il mercato con il rivale iOS di Apple. Sebbene le due piattaforme si rincorrano dal punto di vista delle funzionalità, la più significativa differenza consiste in quella che si chiama apertura, o meglio nelconcetto di ***open***. Mentre la piattaforma di Apple è chiusa (non personalizzabile), il produttore è unico e le tipologie di dispositivi sono in numero limitato, Android rappresenta la collezione di un numero elevatissimo di versioni diverse su dispositivi diversi realizzati da vendor diversi.

Di seguito si riportano molto sinteticamente alcune delle caratteristiche di Android.

Programmato in	C, C++, Java
Famiglia di SO	Unix-like, Linux
Tipo di codice sorgente	Open source
Rilascio iniziale	23 Settembre 2008
Ultimo rilascio stabile	4.1.2 "Jelly Bean" (il 9 Ottobre 2012)
Marketing target	Smartphones, Tablet computers
Piattaforme supportate	ARM, MIPS, x86
Tipo di Kernel	Monolititico (kernel Linux modificato)
Interfaccia utente di default	Grafica (Multi-touch)
Website ufficiale	www.android.com

Android è un SO Linux-based progettato per dispositivi mobili touchscreen, quali gli smartphone e i tablet computer. Esso è attualmente sviluppato da Google congiuntamente con l'Open Handset Alliance. Inizialmente sviluppato da Android Inc, che Google sostenne finanziariamente e successivamente acquistò nel 2005, Android è stato inaugurato nel 2007 con la fondazione della Open Handset Alliance, un consorzio di 86 società di hardware, software e telecomunicazione impegnate a far progredire gli standard aperti per dispositivi mobili.

Google rilascia il codice del SO Android come open source, sotto la licenza Apache. L'Android Open Source Project (AOSP), guidato da Google, ha assunto l'onere di

mantenere e sviluppare ulteriormente Android. Inoltre Android ha una vasta comunità di sviluppatori impegnati a scrivere applicazioni ("app") che estendono la funzionalità dei vari dispositivi e che sono scritte essenzialmente in un'apposita versione di Java. Le app sono disponibili per download attraverso il sito Google Play oppure i siti di terze parti: nell'ottobre 2012, le app ammontavano approssimativamente a 700.000 e il numero stimato di *applications download* da Google Play (e dall'ormai defunto Android Market) è stato di 25 miliardi.

Il primo telefono Android è stato venduto nell'ottobre 2008, ed entro la fine del 2010 Android era diventato la piattaforma per smartphone leader a livello mondiale (superando Symbian che deteneva il record precedente). Ha raggiunto una quota del 75% del mercato degli smartphone in tutto il mondo nel corso del terzo trimestre del 2012, con 500 milioni di dispositivi attivati e 1,3 milioni di attivazioni al giorno. Il SO è anche andato oltre i telefoni cellulari e tablet: attualmente televisori, netbook e macchine fotografiche sono alcuni dei tipi di dispositivi su cui è istallato Android.

2. Storia

Android Inc. è stata fondata a Palo Alto, in California, nell'ottobre 2003 da Andy Rubin e altri partners, con l'intento di sviluppare, nelle parole di Rubin, "... i dispositivi mobili intelligenti che sono più consapevoli della posizione del suo proprietario e le preferenze . ". Nonostante gli evidenti successi passati dei fondatori e dei primi dipendenti, Android Inc. ha inizialmente operato in segreto, rivelando solo che stava lavorando sul software per i telefoni cellulari. Nello stesso anno, Rubin esaurisce le risorse disponibili. Steve Perlman, un amico intimo di Rubin, gli fornisce 10.000 dollari in contanti in una busta, rifiutando una partecipazione nella società.

Google acquisisce Android Inc. il 17 agosto del 2005, che diviene una consociata interamente controllata da Google. Dipendenti chiave di Android Inc., tra cui Rubin, rimangono nell'azienda dopo l'acquisizione. Non si sapeva molto su Android Inc., ma molti all'epoca ipotizzano che Google stia progettando di entrare nel mercato della telefonia mobile con questa mossa. Il team guidato da Rubin sviluppa una piattaforma per dispositivi mobili basata sul kernel di Linux. Google commercializza la piattaforma per produttori di cellulari e operatori con la promessa di dotarla di un sistema flessibile e facilmente aggiornabile. Google può contare sulla collaborazione di una serie di partner produttori di componenti hardware e software e rende noto ai carrier che è disponibile a vari gradi e forme di cooperazione da parte loro.

La speculazione circa l'intenzione di Google di entrare nel mercato delle comunicazioni mobili ha continuato a crescere fino al dicembre 2006. Relazioni della BBC e del "The Wall Street Journal" riportavano che Google volesse dare forte impulso alla sua ricerca e alle applicazioni sui telefoni cellulari. Stampa e media on-line riferivano voci che Google stava sviluppando un portatile di brand Google. Alcuni ipotizzavano che, dato che Google stava definendo specifiche tecniche, essa intendesse presentare prototipi di cella a produttori di telefoni e operatori di rete. Nel settembre 2007, Information Week riportava uno studio Evalueserve secondo cui Google aveva presentato diverse domande di brevetto nel settore della telefonia mobile.

Il 5 novembre 2007, la Open Handset Alliance (un consorzio di aziende tecnologiche tra cui Google, produttori di dispositivi come HTC e Samsung, vettori wireless come Sprint Nextel e T-Mobile, produttori di chipset Qualcomm e Texas Instruments) manifestava l'obiettivo di sviluppare standard aperti per dispositivi mobili. Quel giorno, Android veniva presentato come primo prodotto del consorzio, una piattaforma per dispositivo mobile, costruita sulla versione del kernel Linux 2.6.

Il primo telefono disponibile in commercio per eseguire Android era l'HTC Dream, pubblicizzato il 22 ottobre, 2008.

Dal 2008, Android ha visto susseguirsi numerosi aggiornamenti, che hanno migliorato in modo incrementale il sistema operativo, con l'aggiunta di nuove funzionalità e correzioni dei bug delle versioni precedenti. A ogni rilascio principale viene assegnato in ordine alfabetico il nome di un dessert o delizia zuccherina; per esempio, la versione 1.5 *Cupcake* è stata seguita dalla 1.6 *Donut*. L'ultima versione è la 5.0.1 *Lollipop*, rilasciata il 2 dicembre 2014. In Appendice si riporta una lista cronologica delle versioni di Android rilasciate tra il 23 settembre 2008 e tutto il dicembre 2014.

Nel 2010, Google ha lanciato la sua serie Nexus di dispositivi - una linea di smartphone e tablet con sistema operativo Android, costruita da un partner produttore. HTC ha collaborato con Google per rilasciare il primo smartphone Nexus, il Nexus One. La serie è stata da allora aggiornata con nuovi dispositivi, come i telefoni Galaxy Nexus e Nexus 7 tablet, realizzati da Samsung e Asus, rispettivamente. Google rilascia i telefoni e i tablet Nexus come veicoli di punta per dimostrare tutte le funzionalità software e hardware di Android.

3. L’interfaccia

La schermata iniziale di un HTC Evo 4G, che mostra la *barra di stato*, i *widget* orologio e meteo, e diverse *app*(lication) scorciatoie.

L'interfaccia utente di Android si basa sulla manipolazione diretta, con input tattili che vagamente corrispondono ad azioni del mondo reale, come la *strisciata*, il *tocco*, i *pizzicotti* e i *pizzicotti invertiti* per manipolare gli oggetti sullo schermo. La risposta agli input dell'utente è progettata per essere immediata e fornisce un’interfaccia fluida. Una serie di hardware interno, quali accelerometri, giroscopi e sensori di prossimità, sono utilizzati da alcune applicazioni per rispondere alle azioni da parte dell'utente (per esempio, la regolazione dello schermo da verticale a orizzontale a seconda di come il dispositivo è orientato) o che consentono all'utente di guidare un veicolo ruotando il dispositivo, in un gioco che simula il controllo di un volante in una corsa.

I dispositivi gestiti da Android iniziano caricando (*bootstrap*) la *schermata iniziale*, il punto centrale di navigazione primaria e d’informazioni sul dispositivo, che è simile al desktop disponibile sul PC. Le schermate iniziali Android sono in genere costituite da *icone* delle applicazioni e dei *widget* (deriva dalla contrazione dei termini "window" e "gadget"); le icone delle applicazioni servono per lanciare l'applicazione associata, mentre i widget sono componenti grafici che hanno lo scopo di facilitare all'utente l'interazione con il programma stesso tramite visualizzazioni *live* come il monitoraggio dell'hardware, contenuti ad auto-aggiornamento, come le previsioni del tempo, la casella di posta dell'utente, o semplici collegamenti a cartelle presenti sul computer e news scorrevoli. Una schermata iniziale può essere costituita da più pagine che l’utente può scorrere avanti e indietro.

Presente lungo la parte superiore dello schermo è una *barra di stato*, che mostra le informazioni sul dispositivo e la sua connettività. Questa barra di stato può essere "*tirato*" fino a rivelare una schermata di notifica in cui le applicazioni visualizzano informazioni o aggiornamenti importanti, come ad esempio una e-mail o SMS di testo appena ricevuto, in modo tale da non interrompere immediatamente o disturbare l'utente. Nelle prime versioni di Android tali notifiche venivano sfruttate per aprire l'applicazione in questione, ma gli aggiornamenti più recenti hanno fornito funzionalità

avanzate, come la possibilità di richiamare un numero direttamente dalla notifica di chiamata persa, senza dover prima aprire l'applicazione di chiamata.

Le notifiche sono persistenti fino alla lettura o il rifiuto da parte dell'utente.

4. Le app(lication)

Play Store nel Galaxy Nexus

Android ha una selezione sempre crescente di applicazioni di terze parti, che possono essere acquistate dagli utenti tramite un app(lication) store come Google Play o Amazon Appstore, o scaricando e installando il file APK dell'applicazione da un sito di terze parti. La applicazione *Play Store* permette agli utenti di navigare, scaricare e aggiornare le applicazioni pubblicate da Google e da terze parti sviluppatrici, ed è pre-installata sui dispositivi che rispondono ai requisiti di compatibilità di Google. L'applicazione filtra l'elenco delle applicazioni disponibili che sono compatibili con il dispositivo dell'utente, e gli sviluppatori possono limitare le loro applicazioni a particolari supporti o paesi per motivi di lavoro. Gli acquisti di applicazioni indesiderate possono essere rimborsati entro 15 minuti del momento del download. A partire da settembre 2012, ci sono state più di 675.000 applicazioni disponibili per Android, e il numero stimato di applicazioni scaricate dal Play Store è stato di 25 miliardi.

Le applicazioni sono sviluppate nel linguaggio Java, utilizzando il kit di sviluppo software Android (SDK). Tale kit include un set completo di strumenti di sviluppo, tra cui un debugger, librerie software, un emulatore portatile, documentazione, codice di esempio, e tutorial. L'ambiente di sviluppo integrato ufficialmente supportato (IDE) è Eclipse utilizzando il plugin Android Development Tools (ADT). Altri strumenti di sviluppo sono a disposizione, tra cui un kit di sviluppo nativo per le applicazioni o le estensioni in C o C++ (Google App Inventor), un ambiente visivo per i programmatori alle prime armi, e vari contesti (framework) cross-platform per applicazioni web mobili.
Al fine di aggirare le limitazioni ai servizi Internet di Google nella Repubblica popolare

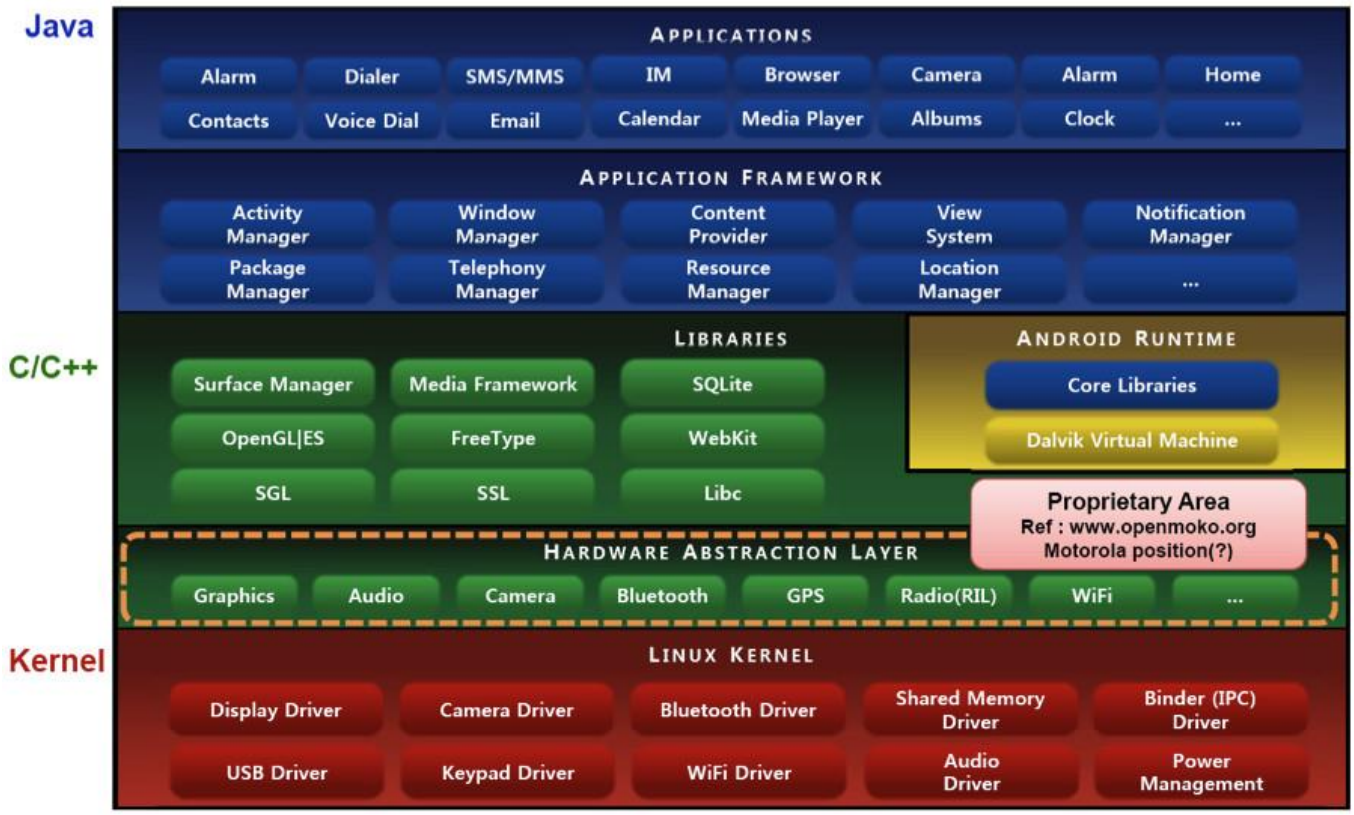
cinese, i dispositivi Android venduti nella RPC sono generalmente personalizzati per utilizzare i servizi di Stato alternativi.

5. La piattaforma Android

La *piattaforma hardware* principale per Android è l'architettura ARM (precedentemente Advanced RISC Machine, prima ancora Acorn RISC Machine), una famiglia di microprocessori RISC a 32-bit sviluppata da ARM Holdings e utilizzata in una moltitudine di sistemi embedded. Grazie alle sue caratteristiche di basso consumo (rapportato alle prestazioni), l'architettura ARM domina il settore dei dispositivi mobili dove il risparmio energetico delle batterie è fondamentale. Attualmente la famiglia ARM copre il 75% del mercato mondiale dei processori a 32 bit per applicazioni embedded, ed è una delle più diffuse architetture a 32 bit del mondo. I processori ARM vengono utilizzati in cellulari, tablet, lettori multimediali, videogiochi portatili, palmari e periferiche per computer (come router, hard disk di rete ecc).

Il supporto per piattaforme x86 riviene dal progetto x86 di Android, e Google TV utilizza una speciale versione x86 di Android.

Per quanto riguarda la *piattaforma software*, la famosa immagine mostrata nella figura seguente ne descrive l'architettura.



Architettura di Android (http://www.gurubee.net/).

Come è possibile immediatamente constatare, la piattaforma Android è costituita da: un

kernel di sistema operativo basato sul Linux 2.6 e Linux 3.x (da Android 4.0 in poi) e completato da un **Hardware Abstraction Layer** (**HAL**); **middleware, librerie e API** scritte in C; software applicativo in esecuzione su un **framework applicativo** che comprende librerie compatibili con Java basata su Apache Harmony.

L'**Hardware Abstraction Layer** (**HAL**) comprende un insieme di funzioni che tengono conto di tutte le differenze fra dispositivi fisici diversi, mascherandole al programma che farà uso di tali dispositivi. Dotando un programma di un HAL se ne migliora la portabilità su altri tipi di computer/sistemi operativi e la funzionalità con dispositivi diversi, perché eventuali modifiche e adattamenti vanno fatti solamente nell'HAL senza toccare il codice del programma stesso; inoltre è relativamente facile aggiungere, all'occorrenza, una sezione all'HAL per gestire un dispositivo che non era stato inizialmente previsto. I programmi, infatti, non accedono mai, per esempio, alla memoria della scheda grafica quando devono modificare l'immagine mostrata sullo schermo. I programmi comunicano al sistema operativo le operazioni da compiere e il sistema operativo provvede a effettuare le modifiche necessarie. Questo consente di modificare l'hardware preposto alla visualizzazione senza dover modificare tutti i programmi. Basta modificare lo strato che accede all'hardware: questo comunemente viene chiamato ***driver***.

L'**ANDROID RUNTIME** è un componente vitale nello stack Android ed è costituito da due parti fondamentali: • La macchina virtuale Dalvik • Le *core libraries*.

La **macchina virtuale Dalvik,** l'elemento forse più importante per chi sviluppa applicazioni, effettua la compilazione *just-in-time* per l'esecuzione di Dalvik dex-code (*Dalvik Executable*), che di solito è tradotto da Java bytecode. Essa è costruita sui servizi offerti dal kernel e dalle diverse librerie native. Opera su dispositivi con CPU e RAM limitate (250-500 MHz) (20-40 MB); non richiede alcuno spazio di swap; impiega una potenza limitata.

Le **core libraries** sono l'altro elemento importante dell'Android runtime. Ciò che l'architettura non mette in evidenza è il ruolo di Java. Nell'architettura Android non esiste

alcuna virtual machine Java e non viene eseguito alcun bytecode Java. In realtà un'applicazione Android si sviluppa in Java, viene compilata in un insieme di risorse e bytecode Java, che viene poi trasformato in bytecode Dalvik attraverso l'applicazione di una serie di ottimizzazioni che lo rendono più compatto ed efficiente a runtime: così, da un'applicazione Java si ottiene un bytecode non Java che viene eseguito in una virtual machine non Java.

Senza pretesa di descrivere qui tutti i componenti della piattaforma precedente (una più dettagliata descrizione, anche se non esaustiva, è riportata nella successiva Appendice 2), possiamo in conclusione osservare che ci sono sostanzialmente tre diverse parti:

- il sistema operativo;
- le librerie;
- l'application framework.

La prima contiene la parte dell'architettura di competenza dei costruttori, i quali dovranno adattare il sistema operativo con il kernel Linux di riferimento e quindi realizzare i diversi driver per l'interazione con il particolare hardware.

La seconda parte è quella delle librerie e ha un'importantissima caratteristica: non sono sviluppate in Java ma perlopiù in C/C++. Le applicazioni Android, infatti, si sviluppano (principalmente) in Java, ma la maggior parte del codice che verrà eseguito sarà codice nativo in C/C++. Questo per il semplice fatto che molti dei componenti della piattaforma sono semplicemente delle interfacce Java di componenti nativi.

Infine, il **framework applicativo** è un toolkit di uso generale che tutte le applicazioni utilizzano: scritto in Java garantisce il riuso e la sostituzione dei componenti.

Per quanto attiene, infine, il layer delle **Applications**, corre l'obbligo di osservare come una delle principali caratteristiche di Android sia quella di permettere la realizzazione di applicazioni attraverso gli stessi strumenti utilizzati per creare la piattaforma stessa. Inoltre, da ormai molto tempo, Android permette lo sviluppo in modo completamente nativo, attraverso un proprio ambiente chiamato NDK (*Native Development Kit*).

La Dalvik Virtual Machine (DVM)

Si tratta di una VM progettata e realizzata da Dan Bornstein con il principale obiettivo di essere ottimizzata per dispositivi mobili. Essa è per esempio in grado di eseguire più processi contemporaneamente, attraverso una gestione ottimale delle risorse condivise. Una delle ottimizzazioni più conosciute riguarda il fatto che si tratta di una VM register-based, a differenza della KVM che è invece la macchina virtuale stack-based di Java, utilizzata dai cellulari per l'esecuzione delle MIDlet ovvero delle applicazioni per dispositivi con caratteristiche congruenti con il *Mobile Information Device Profile* (MIDP).

Attraverso un utilizzo intelligente dei registri di sistema, la DVM permette una maggiore ottimizzazione della memoria in dispositivi con bassa capacità (CPU lenta e scarsa memoria centrale). Si tratta inoltre di una VM ideale per l'esecuzione contemporanea di più istanze, sfruttando la gestione di processi, memoria e thread del sistema operativo sottostante. La Dalvik VM non esegue bytecode Java, ma un codice che si può ottenere da esso e che prende il nome di Dalvik bytecode. Le applicazioni per Android si sviluppano in Java (sfruttando i tool di sviluppo classici come *Eclipse*, *NetBeans* e ora *Android Studio*) per poi trasformare il bytecode Java in Dalvik bytecode. Su un dispositivo Android non verrà eseguito alcun bytecode Java ma un bytecode le cui specifiche sono descritte dal formato DEX (*Dalvik EXecutable*). Le core library e tutte le API a disposizione nella realizzazione delle applicazioni per Android saranno quindi classi Java.

6. LINUX

Il kernel di Android è il risultato di una fork del kernel Linux, a cui sono state applicate modifiche architetturali da parte di Google. Android non ha un sistema X Window nativo di default né supporta il set completo di librerie standard GNU, e ciò rende difficile il porting o adattamento di applicazioni o librerie Linux esistenti. Ma il sostegno di semplici applicazioni C e SDL è possibile mediante l'adozione di un piccolo strato Java e l'utilizzo del JNI.

Alcune nuove funzionalità che Google ha trasferito al kernel Linux (in particolare la funzione di risparmio energetico chiamata *wakelocks*), sono state respinte dai manutentori delle parti principali del kernel, perché Google non ha mostrato alcuna intenzione di mantenere il proprio codice in relazione ai cambiamenti apportati al kernel Linux.

Linux ha incluso le funzionalità *autosleep* e *wakelocks* di Android nel kernel 3.5, dopo molti tentativi precedenti. Le interfacce sono le stesse, ma l'implementazione guida di Linux permette due diverse modalità di sospensione: a memoria (la tradizionale sospensione che utilizza Android), e su disco (ibernazione, come viene chiamata sul desktop). Nel mese di agosto 2011,. Linus Torvalds ha presagito che "alla fine Android e Linux dovrebbero tornare a un kernel comune, ma probabilmente ciò non avverrà prima di quattro o cinque anni". Nel dicembre 2011, Greg Kroah-Hartman, l'attuale responsabile del ramo STABLE del kernel Linux, ha annunciato l'avvio del progetto "Android mainlining", che mira a mettere alcuni driver, patch e caratteristiche Android nel kernel Linux, a partire dal Linux 3.3.

La memoria flash sui dispositivi Android è divisa in diverse partizioni, quali "***/system***" per il sistema operativo e "***/dati***" per i dati utente e le installazioni di applicazioni. A differenza delle distribuzioni Linux per desktop, i proprietari di dispositivi Android non hanno accesso alla *root* del sistema operativo e le partizioni sensibili come ***/system*** sono di sola lettura. Tuttavia, l'accesso alla root può essere ottenuto sfruttando falle nella sicurezza in Android, approccio che viene utilizzato frequentemente dalla comunità open source per migliorare le capacità dei loro dispositivi, ma anche dai malintenzionati per installare virus e malware.

7. Memory management

Poiché i dispositivi Android sono in genere alimentati a batteria, Android è progettato per gestire la memoria (RAM) mantenere il consumo di energia al minimo. Questo contrasta con i sistemi operativi desktop, che generalmente assumono di essere collegati alla rete elettrica e fanno uso a piacimento dell'energia. Quando un'applicazione Android non è più in uso, invece, il sistema la sospende automaticamente nella memoria - mentre l'applicazione è ancora tecnicamente "aperta". Le applicazioni sospese non consumano risorse (in particolare, l'energia della batteria) e sono ferme in background fino a quando sono di nuovo necessarie. Questo ha il duplice vantaggio di aumentare la reattività generale del dispositivo, dal momento che le applicazioni non hanno bisogno di essere chiuse e riavviate ogni volta, ma anche di garantire, rimanendo in background, di non sprecare energia inutilmente.
Android gestisce le applicazioni presenti in memoria automaticamente: quando la memoria è insufficiente, il sistema inizierà uccidendo applicazioni e processi che sono stati inattivi per un po', in ordine inverso rispetto all'ultimo periodo utilizzato (cioè che sono stati meno recentemente attivi). Questo meccanismo è stato progettato per essere invisibile per l'utente, in modo che gli utenti non abbiano bisogno di gestire la memoria o l'uccisione delle applicazioni stesse. La non precisa conoscenza del criterio anzidetto di gestione della memoria ha indotto, sul Google Play Store, un'esorbitante e inutile offerta di app per la gestione ottimale della RAM. App come quelle citate sono generalmente considerate, in Android, fare più male che bene.

8. Pianificazione degli aggiornamenti

Da sinistra a destra: HTC Dream (G1), Nexus One, Nexus S, Galaxy Nexus

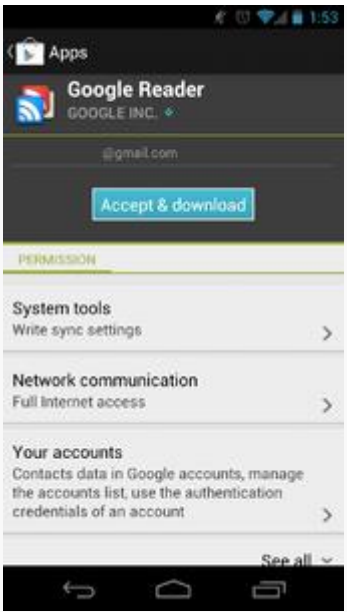
Google fornisce aggiornamenti importanti ad Android, tipicamente incrementali, ogni sei-nove mesi. La maggior parte dei dispositivi è in grado di riceverli via etere. L'ultimo aggiornamento importante è Android 4.2 Jelly Bean.
Rispetto ai sistemi operativi mobili rivali, ovvero in particolare iOS, gli aggiornamenti Android sono in genere molto lenti nel raggiungere i dispositivi. Spesso sono necessari diversi mesi dalla data di rilascio ufficiale di Google perché effettivamente tali aggiornamenti siano distribuiti ai dispositivi. Ciò è dovuto in parte alla vasta varietà dell'hardware dei dispositivi Android (per i quali ogni aggiornamento deve essere su misura). Infatti il codice sorgente ufficiale di Google funziona correttamente solo sui telefoni di punta Nexus. L'adattamento (*porting*) di Android a ciascun hardware specifico è un processo che richiede, per i produttori di dispositivi, non trascurabili tempi e risorse. Per di più i produttori danno priorità al porting sui loro dispositivi più recenti e spesso lasciano indietro quelli più vecchi. Quindi, gli smartphone più vecchi sono spesso non aggiornati se il produttore decide che non ne vale la pena, a prescindere dal fatto che il telefono sia in grado di essere aggiornato. Questo problema è aggravato, per i produttori, dall'impegno ulteriore di apportare gli aggiornamenti Android anche alla propria interfaccia e alle proprie applicazioni, e ciò deve deve essere ripetuto per ogni nuovo rilascio. Alcuni commentatori hanno notato che i produttori hanno addirittura un incentivo finanziario a non aggiornare i loro dispositivi, poichè la mancanza di aggiornamenti per i dispositivi esistenti spinge la clientela all'acquisto di quelli più recenti. Ulteriori ritardi possono essere introdotti dagli operatori wireless che, dopo aver ricevuto gli aggiornamenti da parte dei produttori, personalizzano e adattano il marchio Android ai loro bisogni e conducono test approfonditi sulle loro reti prima di inviare l'aggiornamento agli utenti.

9. Open source community

Android ha una comunità attiva di sviluppatori e appassionati che utilizzano il codice sorgente di Android per sviluppare e distribuire le proprie versioni modificate del sistema operativo. [71] Queste comunità hanno sviluppato vari rilasci, il più diffuso dei quali è CyanogenMod, e spesso apportano nuove funzionalità e aggiornamenti ai dispositivi più velocemente che attraverso i canali ufficiali produttore/carrier, pur senza effettuare accurati e intensi test o assicurare garanzia di qualità: spesso i rilasci contengono modifiche non idonee per utenti non tecnici.
Storicamente, le prime reazioni dei produttori di tablet e smartphone e degli operatori di telefonia mobile sono state tipicamente non collaborative circa lo sviluppo di firmware di terze parti. I produttori hanno espresso preoccupazione per il cattivo funzionamento dei dispositivi che eseguono software non ufficiale e circa i costi di supporto derivanti.
D'altra parte, i firmware modificati, come CyanogenMod, a volte offrono funzioni (ad esempio, il tethering) per le quali i vettori sono disposti a pagare generosamente. Come risultato, gli ostacoli tecnici, tra cui il blocco del bootloader bloccato e l'accesso limitato ai permessi di root, sono stati frequenti in molti dispositivi.

Tuttavia, mano a mano che il software sviluppato dalle comunità è divenuto più popolare, e in seguito ad una dichiarazione della Biblioteca del Congresso negli Stati Uniti, che permette la violazione dei vincoli ("*jailbreaking*") ai dispositivi mobili, i produttori e carrier (tra cui HTC, Motorola, Samsung e Sony Ericsson) hanno ammorbidito la loro posizione in materia di sviluppo di terze parti, offrendo un po' di sostegno e promozione a tale tipo di sviluppo. Nel tempo, quindi, la necessità di aggirare le restrizioni hardware per installare firmware ufficiale, è diminuita e un numero crescente di dispositivi viene fornito con bootloader sbloccati o sbloccabili, come nel caso della serie Nexus di telefoni, anche se di solito richiede che gli utenti rinuncino alla garanzia sui loro dispositivi per farlo.

10. Security and privacy



Autorizzazioni App nel Play Store.

Le applicazioni Android vengono eseguite in una "*sandbox*", una zona isolata del sistema che non ha accesso al resto delle risorse del sistema, a meno che i permessi di accesso siano concessi esplicitamente dall'utente quando si installa l'applicazione. Prima di installare un'applicazione, Play Store visualizza tutte le autorizzazioni necessarie: per un gioco può essere necessario, per esempio, attivare la vibrazione o salvare i dati su una scheda SD, ma non dovrebbe aver bisogno di leggere i messaggi SMS o accedere alla rubrica. Dopo aver esaminato queste autorizzazioni, l'utente può decidere se installare l'applicazione.
Il sistema di *sandboxing* e la necessità delle autorizzazioni riduce l'impatto delle vulnerabilità e dei bug nelle applicazioni, ma la confusione dello sviluppatore e la documentazione limitata hanno portato ad applicazioni che richiedono autorizzazioni non necessarie, riducendo l'efficacia del sandboxing. Diverse imprese di sicurezza, come la Lookout Mobile Security, AVG Technologies, e McAfee, hanno rilasciato un software antivirus per i dispositivi Android. Questo software è inefficace poiché il *sandboxing* si applica anche alle applicazioni di tali società, limitando la loro capacità di eseguire una scansione accurata delle minacce.

La ricerca della società di sicurezza Trend Micro riferisce che il tipo più comune di malware per Android è costituito dall'abuso dell'utilizzo di servizi a pagamento: messaggi di testo vengono inviati dai telefoni infetti a numeri telefonici a tariffazione, senza il consenso o addirittura in maniera sconosciuta all'utente. Altri malware introducono videate indesiderate e pubblicità intrusive sul dispositivo, oppure inviano informazioni personali a terzi non autorizzati. Le minacce alla sicurezza su Android vanno crescendo

esponenzialmente; tuttavia Google sostiene che la minaccia di malware e virus su Android è esagerata dalle società di sicurezza per ragioni commerciali, e accusa il settore della sicurezza di giocare sulle paure per vendere il software di protezione antivirus, sostenendo che il malware pericoloso in realtà è estremamente raro.

Google sta sviluppando uno scanner di malware per il suo Play Store, destinato a evidenziare applicazioni sospette e avvisare gli utenti di potenziali problemi con un'applicazione, prima di scaricarla. La versione Android 4.2 (*Jelly Bean*) è stata rilasciata nel 2012 con funzioni di sicurezza avanzate, tra cui: uno scanner di malware integrato nel sistema, che funziona in combinazione con Google Play, ma può esplorare anche applicazioni installate da fonti terze; e un sistema di allarme che avvisa l'utente quando un app tenta di inviare un messaggio di testo a pagamento, bloccando il messaggio a meno che l'utente lo autorizzi esplicitamente.
Gli smartphone Android hanno la capacità di segnalare la posizione dei punti di accesso Wi-Fi, rilevati via via che gli utenti di telefonia si muovono, al fine di costruire banche dati contenenti le posizioni fisiche di centinaia di milioni di tali punti di accesso. Questi database formano mappe elettroniche per individuare gli smartphone, consentendo loro di eseguire applicazioni come Foursquare, Google Latitude, Facebook Places, e per offrire annunci basati sulla localizzazione. Infine software di monitoraggio di terze parti come TaintDroid, risultato di un progetto accademico di ricerca, può, in alcuni casi, rilevare quando le informazioni personali vengono inviate dalle applicazioni ai server remoti.

11. Licenze

Il codice sorgente di Android è disponibile con licenze di software libero e open source. Google pubblica la maggior parte del codice (tra cui la pila di software di rete e di telefonia) sotto la versione di Apache License 2.0, e il resto, le modifiche al kernel Linux, sotto la versione GNU General Public License 2. L'Open Handset Alliance sviluppa le modifiche al kernel di Linux pubblico, con il codice sorgente disponibile pubblicamente in ogni momento. Il resto di Android è sviluppato in privato da parte di Google, con il codice sorgente che viene rilasciato pubblicamente quando viene rilasciata una nuova versione. Tipicamente Google collabora con un produttore di hardware per la produzione di una nuova versione di Android per un dispositivo di bandiera (parte delle "*Google Nexus series*"), quindi rende il codice sorgente disponibile dopo che tale dispositivo è stato rilasciato.
All'inizio del 2011, Google ha scelto di trattenere temporaneamente il codice sorgente di Android alla versione 3.0 *Honeycomb* per solo tablet. La ragione, resa nota da Andy Rubin in un *post* sul blog ufficiale di Android, è che Honeycomb è stata adattata per la produzione del dispositivo Xoom della Motorola, e non si voleva che qualche terza parte provocasse una "*brutta impressione*", nel tentativo di mettere su smartphone una versione di Android progettata per tablet. Il codice sorgente è stato ancora una volta reso disponibile nel novembre 2011 con il rilascio di Android 4.0.

Anche se il software è open-source, i produttori di dispositivi non possono utilizzare il marchio Android di Google, a meno che Google certifichi che il dispositivo è conforme con la loro *Document Compatibility Definition* (CDD). I dispositivi devono anche soddisfare questa definizione per essere ammessi a concedere in licenza le applicazioni *closed-source* di Google, tra cui Google Play. Poiché Android non è completamente rilasciato sotto una licenza compatibile GPL (si tenga conto che il codice di Google è sotto la licenza Apache) e anche perché Google Play consente software proprietario, Richard Stallman e la Free Software Foundation hanno criticato Android e hanno raccomandato l'uso di alternative come asReplicant.

12. Accoglienza

Android ha ricevuto una reazione tiepida quando è stato lanciato nel 2007. Anche se gli analisti sono stati colpiti dalla rilevanza delle aziende tecnologiche che avevano collaborato con Google per formare la Open Handset Alliance, non era chiaro se i produttori di cellulari sarebbero stati disposti a sostituire i loro sistemi operativi esistenti con Android. L'idea di una piattaforma di sviluppo open source, basata su Linux, suscitava interesse, ma c'erano perplessità su Android a causa: della forte concorrenza di aziende affermate nel mercato degli *smartphone*, come Nokia e Microsoft, e dei sistemi operativi mobili, rivali di Linux, che erano in fase di sviluppo. Queste aziende affermate erano scettiche: si attribuisce a Nokia la citazione: "non vediamo questo come una minaccia," e a un membro del team di Microsoft Windows Mobile la dichiarazione: "non vedo l'impatto che hanno intenzione di avere ".
Da allora Android è cresciuto fino a diventare il sistema operativo per smartphone più diffuso e "uno delle più veloci esperienze mobili disponibili." I media e i critici hanno evidenziato la natura *open source* del sistema operativo come uno dei suoi decisivi punti di forza, che hanno permesso ad aziende come Amazon, Barnes and Noble, Ouya, Baidu, di derivare (*fork*) dalla versione "open source" e da hardware rilasciato la propria versione personalizzata di Android. Di conseguenza, Android è stato descritto dal sito web tecnologico *Ars Technica* come "*praticamente il sistema operativo predefinito per il lancio di nuovo hardware*" per le aziende senza proprio piattaforme mobili. Questa apertura e flessibilità sono presenti anche a livello di utente finale: Android, infatti, consente un'ampia personalizzazione dei dispositivi e le applicazioni sono disponibili gratuitamente da *app store* non Google e siti web di terze parti. Questi sono stati citati come tra i principali vantaggi dei telefoni Android.

Nonostante il suo successo su smartphone, l'adozione di Android sui tablet è stata lenta. La colpa è principalmente dovuta al paradosso aristotelico dell'uova e della gallina: i consumatori sono riluttanti ad acquistare un tablet Android a causa della mancanza di applicazioni tablet di alta qualità, ma gli sviluppatori sono restii a spendere tempo e risorse per sviluppare applicazioni tablet fino a quando non c'è un mercato significativo dei tablet. Altri fattori che hanno determinato tale lenta adozione sono gli alti prezzi dei tablet e il predominio di Apple iPad. Tale situazione è cominciata a cambiare nel 2012 con il rilascio di Nexus 7 e con la spinta di Google agli sviluppatori per scrivere applicazioni tablet migliori.

13. Quota di mercato e tasso di adozione

La società di ricerca Canalys stimava nel secondo trimestre del 2009 che Android avesse una quota del 2,8% delle vendite di smartphone in tutto il mondo. Con il quarto trimestre del 2010 la quota era cresciuta al 33%, diventando così Android la piattaforma per smartphone più venduta. Nel terzo trimestre del 2011 Gartner ha stimato che più della metà (52,5%) del mercato degli smartphone appartiene ad Android. Con il terzo trimestre del 2012 Android ha, secondo la ricerca società IDC, una quota del 75% del mercato globale degli smartphone.
Nel luglio 2011, Google asseriva che 550.000 nuovi dispositivi Android venivano attivati ogni giorno (a partire dai 400.000 al giorno di maggio), e che più di 100 milioni di dispositivi erano stati attivati, con il 4,4% di crescita a settimana. Nel settembre del 2012, 500 milioni di dispositivi erano stati attivati con 1,3 milioni di attivazioni al giorno.
La capacità di Android di conseguire una quota di mercato considerevole è stato attribuita in parte alla strategia di Google di rilasciare prontamente Android ai produttori di dispositivi di fascia bassa.

14. Appendice 1

La seguente tabella mostra, ad oggi, i principali aggiornamenti del sistema operativo Android, elencati in ordine cronologico per livelli della loro *Application Programming Interface* (API) ufficiale.

Versione	Data di pubblicazione	Caratteristiche
pre-1.0 Astro Boy, Bender	//	Release sperimentali precedenti alla 1.0, contrassegnate come "milestone" e indicate con un numero di serie.
1.0	23 settembre 2008	API Level 1. Il primo telefono commerciale a montare questo sistema operativo è stato l'HTC Dream (noto come T-Mobile G1 negli Stati Uniti).
1.1	9 febbraio 2009	API Level 2. Update per risolvere vari bug (gli utenti HTC Dream TIM sono gli unici ad essere ancora fermi a questa versione).
1.5 Cupcake	30 aprile 2009	API Level 3. Linux kernel 2.6.27. Maggior integrazione con i servizi Google, supporto per i widget.
1.6 Donut	15 settembre 2009	API Level 4. Linux kernel 2.6.29. Aggiunta di ricerca vocale e testuale per i contenuti presenti in locale e sul Web. Introdotta la sintesi vocale e le gesture.
2.0 Eclair	26 ottobre 2009	API Level 5. Linux kernel 2.6.29. Aggiunte numerose funzionalità per la fotocamera. Migliorata la sincronizzazione dell'account Google e aggiunto il supporto agli account Exchange. Aggiunto il supporto al multi-touch e ai live wallpaper. UI e prestazioni migliorate.
2.0.1 Eclair	3 dicembre 2009	API Level 6. Risolti alcuni bug minori.
2.1 Eclair	12 gennaio 2010	API Level 7. Minor release.
2.2 Froyo	20 maggio 2010	API Level 8. Linux kernel 2.6.32. Drastico miglioramento prestazionale, dovuto ad una migliore gestione delle risorse hardware (compilazione JIT). Tethering USB e Wi-Fi. Integrazione del motore JavaScript V8 di Google Chrome nel browser di sistema. Supporto alla tecnologia Adobe Flash. Migliorie apportate a gran parte delle altre applicazioni di sistema.
2.2.1 Froyo	18 gennaio 2011	API Level 8. Incremento prestazionale e miglioramento della sicurezza.

2.2.2 Froyo	22 gennaio 2011	API Level 8. Risolto un bug relativo all'invio degli SMS riscontrato su Nexus One.
2.2.3 Froyo	21 novembre 2011	API Level 8. Patch di sicurezza.
2.3 Gingerbread	6 dicembre 2010	API Level 9. Linux kernel 2.6.35. UI aggiornata per essere più user-friendly. Aggiunto il supporto agli schermi XL (risoluzione WXGA e superiori). Supporto nativo al SIP VoIP e alla tecnologia NFC. Tastiera riprogettata (precisione predittiva aumentata e copia/incolla migliorato). Aggiunta l'app Download Manager, per la gestione unificata di tutti i download effettuati dalle app di sistema e non. Supporto nativo a sensori come giroscopio e barometro. Migliorata la gestione energetica.
2.3.1 Gingerbread	dicembre 2010	API Level 9. Risolti alcuni bug riscontrati nel Nexus S.
2.3.2 Gingerbread	gennaio 2011	API Level 9. Risolti alcuni bug riscontrati nel Nexus S.
2.3.3 Gingerbread	9 febbraio 2011	API Level 10. Numerosi miglioramenti e correzioni nelle API.
2.3.4 Gingerbread	28 aprile 2011	API Level 10. Supporto per la chat video e vocale tramite Google Talk.
2.3.5 Gingerbread	25 luglio 2011	API Level 10. Migliorate le applicazioni Fotocamera e Gmail. Efficienza energetica migliorata. Fix di alcuni bug riscontrati su Nexus S 4G e Galaxy S.
2.3.6 Gingerbread	2 settembre 2011	API Level 10. Risolto un bug relativo alla ricerca vocale.
2.3.7 Gingerbread	21 settembre 2011	API Level 10. Supporto a Google Wallet (solo per Nexus S 4G).
3.0 Honeycomb	22 febbraio 2011	API Level 11. Linux kernel 2.6.36. Versione ottimizzata per tablet. Introdotta nuova UI, denominata "Holo". Aggiunta la barra di sistema (pulsanti software Home, Indietro, Task manager) e la Action Bar (fornisce accesso ad opzioni che variano in base al contesto). Browser multi-tab. Accelerazione hardware e supporto per processori multi-core. Possibilità di criptare tutti i dati personali.
3.1 Honeycomb	10 maggio 2011	API Level 12. Miglioramenti alla UI. Widget ridimensionabili. Supporto per le periferiche USB (flash drive, gamepad).
3.2 Honeycomb	15 luglio 2011	API Level 13. Ampliato il supporto hardware. SDK aggiornato per permettere agli sviluppatori di personalizzare più a fondo la UI.
3.2.1 Honeycomb	20 settembre 2011	API Level 13. Aggiornamento di Google Ricerca libri e Android Market. Bug fixes.

3.2.2 Honeycomb	30 agosto 2011	API Level 13. Risolti alcuni bug riscontrati nel Motorola Xoom 4G.
3.2.3 Honeycomb		API Level 13. Risolti alcuni bug riscontrati nel Motorola Xoom e nel Motorola Xoom 4G.
3.2.4 Honeycomb	dicembre 2011	API Level 13. Supporto per i Tablet 3G e 4G dell'opzione "Pay as You Go".
3.2.5 Honeycomb	gennaio 2012	API Level 13. Risolti alcuni bug riscontrati nel Motorola Xoom e nel Motorola Xoom 4G.
3.2.6 Honeycomb	febbraio 2012	API Level 13. Risolti alcuni bug riscontrati nella disatiivazione della connettività dati in modalità aereo per il Motorola Xoom 4G americano.
4.0 Ice Cream Sandwich	19 ottobre 2011	API Level 14. Linux kernel 3.0.1. UI completamente riprogettata: prestazioni migliorate, pulsanti virtuali al posto di quelli hardware (per i dispositivi che ne sono privi), cartelle più facili da creare, launcher personalizzabile, nuovo font di sistema (Roboto). Aggiornate tutte le app di sistema per sfruttare le nuove API. Possibilità di scattare screenshots integrata nell'OS. Dettatura in tempo reale. Face Unlock, per sbloccare il dispositivo tramite il software di riconoscimento facciale. Possibilità di accedere alle applicazioni direttamente dalla schermata di sblocco. Fotocamera migliorata con: ritardo di scatto nullo (zero shutter lag), modalità panorama e zoom durante la ripresa di video. App "Contatti" con integrazione con i social network. Android Beam (scambio di dati tramite NFC). Wi-Fi Direct.
4.0.1 Ice Cream Sandwich	21 ottobre 2011	API Level 14. Risolti alcuni bug minori riscontrati nel Galaxy Nexus.
4.0.2 Ice Cream Sandwich	28 novembre 2011	API Level 14. Risolti alcuni bug minori riscontrati nel Galaxy Nexus marchiato Verizon Communications.
4.0.3 Ice Cream Sandwich	16 dicembre 2011	API Level 15. Accessibilità migliorata.
4.0.4 Ice Cream Sandwich	29 marzo 2012	API Level 15. Migliorate funzione multitasking, rotazione dello schermo e applicazione Camera.
4.1 Jelly Bean	9 luglio 2012	API Level 16. Linux kernel 3.0.31. Riconoscimento del tocco migliorato, ottimizzato l'utilizzo della CPU, digitazione testo migliorato, voice typing offline, migliorata la gestione dei widget (possibilità di eliminarli con le gesture, ridimensionamento automatico), miglioramenti notevoli nella fluidità grazie a "Project Butter", importanti ottimizzazioni dell'applicazione fotocamera, nuove funzionalità per la condivisione di foto e video tramite NFC, Android Beam migliorato, gesture avanzate per le notifiche, nuovo

		servizio *Google Now*, sintesi vocale migliorata, *Google Play Store* aggiornato, riconoscimento vocale avanzato (GSV). Abbandono ufficiale al supporto della tecnologia Adobe Flash.
4.1.1 Jelly Bean	23 luglio 2012	API Level 16. Risolto un bug sul Nexus 7 (2012) riguardante l'impossibilità di cambiare l'orientamento dello schermo in qualsiasi applicazione.
4.1.2 Jelly Bean	9 ottobre 2012	API Level 16. Correzioni di bug e miglioramenti delle prestazioni, introduzione delle gesture con un dito per espandere/chiudere le notifiche, nuovo pulsante "Seleziona tutto" sulla tastiera Swype, supporto per il Nexus 7 (2012) della rotazione della Lock/home screen.
4.2 Jelly Bean	13 novembre 2012	API Level 17. Linux kernel 3.4.0. Implementazione nativa della funzione swipe nella tastiera, utilizzo delle impostazioni della fotocamera con una sola gesture, introduzione dei toggle per le impostazioni rapide, migliorato il servizio *Google Now*, aggiornato il supporto ad *Android Beam*, introdotta la modalità fotografica a 360° (*PhotoSphere*), supporto per i widget nella *lockscreen*, aggiunto il pinch-to-zoom per l'app *Gmail*, supporto multi-utente per tablet.
4.2.1 Jelly Bean	27 novembre 2012	API Level 17. Risolti alcuni bug causati dall'aggiornamento alla versione 4.2.
4.2.2 Jelly Bean	11 febbraio 2013	API Level 17. Migliorate le prestazioni e la stabilità, risolti i problemi al Bluetooth A2DP per la trasmissione wireless di audio stereo, modificate le icone del Bluetooth e del Wi-Fi nell'area notifiche che diventano anche toggle (tramite pressione prolungata si attivano/disattivano), aggiunto il tempo stimato del download delle applicazioni nella barra delle notifiche.
4.3 Jelly Bean	24 luglio 2013	API Level 18. Introdotte le OpenGL ES 3.0, nuova fotocamera, introdotta la funzione Wi-Fi always-on, dialer con la ricerca dei contatti, accesso alle notifiche da parte delle applicazioni, possibilità di introdurre contenuti DRM, connessione Bluetooth con i dispositivi a basso consumo di energia e introdotte le impostazioni per il multiutente (Restricted Profile) che permettono di scegliere quali contenuti e quali applicazioni gli utenti possono visualizzare.
4.3.1 Jelly Bean	4 ottobre 2013	API Level 18. Risolti alcuni bug riscontrati nel Nexus 7 (2013) LTE.
4.4 KitKat	31 ottobre 2013	API Level 19. Rinnovata l'interfaccia grafica (*Navbar* e *Status bar* trasparenti per sfruttare meglio tutto lo spazio dello schermo) e introdotto il full screen completo. *Hangouts* è diventato il client ufficiale per SMS e MMS, introdotte nuove funzioni di chiamata, aggiunta la possibilità di catturare video in MP4 di ciò che avviene sullo schermo, aggiunto il supporto nativo alla stampa di foto,

		documenti e pagine web verso Google Cloud Print, HP ePrint, e altre stampanti compatibili. Aggiunta una nuova voce dedicata alla Home nelle impostazioni per lo switch tra launcher. Aggiunte le emoji alla tastiera di Google. Rinnovata l'app Download, con opzioni per l'ordinamento e la visualizzazione. Supporto per 3 nuovi tipi di sensore (vettore di rotazione geomagnetica, rilevatore e contatore di passi) e aggiunta la funzione contapassi. Diminuito il consumo di batteria durante la riproduzione audio. Introduzione di ART (Android RunTime), un nuovo compilatore, attualmente in via sperimentale, attivabile dalle Opzioni Sviluppatore (anche se non su tutti i dispositivi). Ottimizzato il funzionamento del sistema sui dispositivi con poca RAM (anche con soli 512 MB). Questa versione di Android era precedentemente conosciuta col nome **Key Lime Pie** e col numero di versione **5.0**. Il nome originario (mai comunque ufficializzato) venne cambiato prima della pubblicazione in seguito ad un accordo tra Google e la Nestlé (l'azienda che produce i Kit Kat), mentre il numero di versione era dovuto solo a dei rumor sbagliati.
4.4.1 KitKat	5 dicembre 2013	API Level 19. Risolti alcuni bug che impedivano la configurazione di account Exchange, introdotte le barre trasparenti nel blocco schermo, sostituita l'app *Galleria* con *Foto* e aggiornato il software della fotocamera; per i Nexus 5, migliorata la velocità di alcune caratteristiche della fotocamera come la messa a fuoco (specialmente in condizioni di scarsa luminosità), il bilanciamento del bianco, la chiusura dell'otturatore e introdotta la possibilità di effettuare *pinch to zoom* sullo smartphone in modalità HDR+.
4.4.2 KitKat	9 dicembre 2013	API Level 19. Correzioni di bug e miglioramenti delle prestazioni vari (compresa la fotocamera), più alcuni fix per la sicurezza. Correzioni sugli indicatori delle *voice mail*.
4.4.3 KitKat	3 giugno 2014	API Level 19. Correzioni di bug e miglioramenti delle prestazioni varie, rinnovo dell'interfaccia dei Contatti e del Dialer per effettuare le chiamate.
4.4.4 KitKat	19 giugno 2014	API Level 19. Aggiornamento minore per la risoluzione di alcuni bug di sicurezza riguardanti OpenSSL.
5.0 Lollipop	3 novembre 2014[97]	API Level 21. UI Completamente rinnovata, soprannominata Material Design, grazie al quale ogni elemento della UI ha un valore di "elevazione" che dice al sistema qual è la sua posizione rispetto a tutti gli altri elementi. 5000 nuove API. Nuovo multitasking. Animazioni a 60 FPS. Importanti miglioramenti della gestione audio. Integrazione di Samsung Knox per la sezione business. Introduzione di Google Fit per il rilevamento delle attività fisiche. Nuovo kernel 3.10.x. Aggiornamento del font *Roboto*. Eliminazione della runtime Dalvik in favore di **ART** (**A**ndroid **R**un**t**ime). Supporto nativo ai 64 bit. Bluetooth 4.1. USB Audio. Burst Mode. Miglioramenti generali sull'autonomia grazie a Project Volta e Battery Saver. Double Tap to Wake, se supportato

		dall'hardware; inoltre, sempre se supportato, il dispositivo dovrebbe risvegliarsi quando lo prendiamo in mano. Netto miglioramento delle prestazioni grafiche grazie al supporto all'OpenGL ES 3.1 e all'accoppiamento con Android Extension Pack. Miglioramento della fotocamera grazie ad API dedicate. Notifiche Heads Up in applicazioni Fullscreen. Personal Unlocking. Aggiunto il multi-utente su smartphone e la modalità ospite. Nuova LockScreen con notifiche (disattivabili) disposte in base alla loro importanza. Importante miglioramento della sicurezza grazie alla versione 6.0 del Google Play Service. Questa versione di Android è stata distrbuita come preview per gli sviluppatori col nome di **Android L** il 26 giugno 2014 e senza numero di versione. Il nome e il numero definitivo della versione è stato ufficializzato il 15 ottobre 2014.
5.0.1 Lollipop	3 dicembre 2014	API Level 21. Aggiornamento minore per la risoluzione di alcuni bug tra cui il factory reset non voluto sbagliando per alcune volte consecutive la password nel lock screen e il miglioramento della riproduzione dei video su Nexus 7 (2013).
5.0.2 Lollipop	19 dicembre 2014	API Level 21. Risolti alcuni bug minori (aggiornamento iniziale per Nexus 7 (2012), e in sequito per tutti gli altri Nexus).

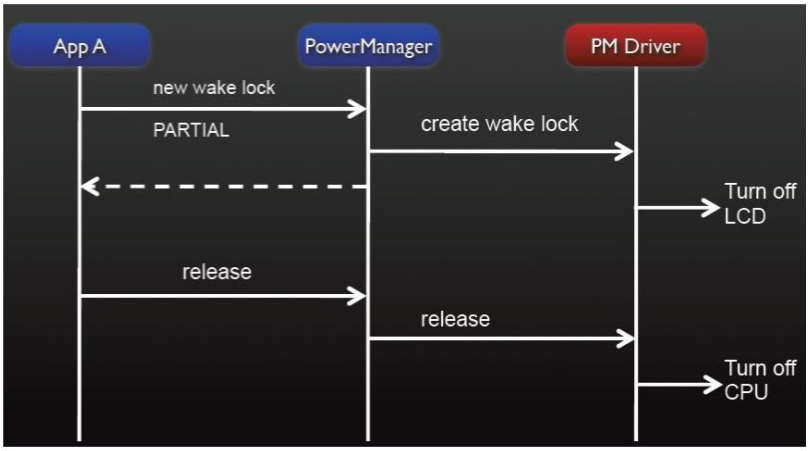
15. Appendice 2

Come anticipato in precedenza, allo **strato del kernel Linux** di Android sono state applicate modifiche architetturali da parte di Google. Alcune delle caratteristiche principali risultanti sono, oltre ai driver di pilotaggio dei dispositivi più comuni,:

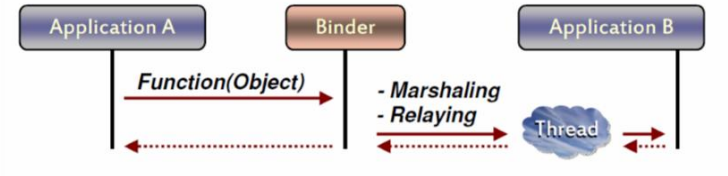
- Un driver (**Power Management**) per la gestione energetica, costruito come una patch di quello standard (PM) utilizzato nei sistemi Linux, che consente alle applicazioni in esecuzione sulla Dalvik machine di utilizzare l'alimentazione con politiche di gestione molto restrittive: la CPU non dovrebbe consumare energia se nessuna applicazione o servizio la richiede. I vari componenti richiedono risorse alla CPU mediante i cosiddetti *wake locks*; se non ci sono wake locks attivi, Android mette in pausa il processore. In realtà si consente alle applicazioni user di inibire il passaggio del sistema allo stato *sleep* o *suspend* mediante il meccanismo delle wake locks.

  Esistono 4 tipi diversi di wake lock:
  • PARTIAL WAKE LOCK: La CPU rimane attiva, anche se viene premuto il pulsante di spegnimento
  • SCREEN DIM WAKE LOCK: lo schermo rimane acceso, ma oscurato
  • SCREEN BRIGHT WAKE LOCK: lo schermo resta acceso con luminosità normale
  • FULL WAKE LOCK : La tastiera e lo schermo rimangono accesi con retro-illuminazione normale
  Solamente il partial wake lock assicura che la CPU sia completamente attiva; gli altri 3 tipi consentono alla CPU di andare in *sleep* quando viene premuto il pulsante di accensione

  Per default il sistema Android tende a mettere il sistema in pausa o in sospensione il più presto possibile. Le wake lock possono essere ottenute sia dalle componenti del kernel che dai processi dello user space.



- Un driver (**Binder IPC Driver**) basato su *OpenBinder* per facilitare la comunicazione tra processi (Inter Process Communication) nella piattaforma.



- Un driver **Shared Memory Driver**, che funziona da *Low Memory Killer* perché, basandosi sui feedback provenienti dalle applicazioni utente, è in grado di terminare i processi per liberare memoria quando necessario.
- Il driver **Android Alarm**, che fornisce ***timer*** che possono risvegliare il dispositivo da uno stato di pausa ed un timer monotono che si avvia quando il dispositivo è attivo.
- **YAFFS**, file system Linux ottimizzato per memorie NAND, dotato delle seguenti caratteristiche:

  • Journaling: file system basato su log, garantisce la robustezza in caso di cadute di alimentazione, necessita di una parte della RAM

  • Garbage Collector (GC): viene richiamato quando lo spazio libero diventa basso e determina il ricorso al meccanismo di *block erasure*.

  • Basso consumo di memoria; Flessibilità: è altamente configurabile e può essere essere realizzato su misura per vari tipi di memorie flash, con differenti opzioni di correzione di errore, caching ecc.; Portabilità: la struttura modulare del codice garantisce il porting su differenti sistemi operativi; Robustezza: Bad Block Management (BBM)

  • Supporto a POSIX: directory, hard link e symbolic link

  • Block Eraser (BE): su memoria NAND non è possibile riscrivere o cancellare un singolo segmento (*chunk*) di un blocco costituito da 64 chunk; per cancellare un segmento in memoria occorre cancellare l'intero blocco: ciascun segmento da cancellare o riscrivere è segnato come "dirty"; quando tutti i segmenti di un blocco sono "dirty" il blocco può essere cancellato. Il Garbage Collector sceglie un blocco, sposta le pagine corrette in un altro blocco, contrassegna il blocco scelto come "dirty", in quanto contenente solo segmenti "dirty", e lo elimina.

La necessità di un simile GC e di un BE deriva dal fatto che, oltre a non essere riscrivibili o cancellabili i singoli segmenti, le memorie flash hanno un numero limitato di cicli di lettura e scrittura, con deterioramento dell'integrità dei dati alla fine del ciclo di vita della memoria, per cui sono indispensabili tecniche di livellamento dell'usura (*wear leveling*), di utilizzo uniforme di intere sezioni per ottimizzare la durata globale della memoria, oltre che di gestione dei blocchi danneggiati (*Bad Block Management*).

Per quanto riguarda lo **strato del middleware, librerie e API** scritte in C/C++, come già riferito nel par. 5, molti dei componenti della piattaforma sono semplicemente delle interfacce Java di componenti nativi.
Tale livello contiene le librerie che forniscono la maggior parte delle funzionalità messe a disposizione da Android e gli sviluppatori possono sfruttare le potenzialità di queste librerie attraverso il livello di application framework. Tra le altre si citano:

- **Bionic**, un sottoinsieme delle librerie di utilità Linux (GNU glibc), adatto per dispositivi portatili Linux-based.
- **SQLite**, potente e leggero DBMS relazionale contenuto in una relativamente piccola libreria C (500 kb). Il suo *engine* non è un processo stand-alone, ma un database SQLite può essere incorporato in un altro programma. Il database è memorizzato come un singolo file multi-piattaforma sulla macchina host e l'applicazione può accedere a SQLite attraverso semplici e convenzionali chiamate a funzioni.
- **Surface Manager**, responsabile della composizione delle superfici di disegno su display mobili; gestisce l'accesso dei differenti processi al sottostante sistema per la grafica 2D e 3D; le superfici sono passate come buffer attraverso chiamate del componente *OpenBinder (IPC)*;
- **SGL**, una libreria grafica 2D specializzata per Android.
- **Media Framework**, fornito dal package multimediale *OpenCore* di PacketVideo, offre una struttura universale per applicazioni multimediali mobili e supporta i formati standard per il video e l'audio (MPEG4, MP3, AAC, AMR, JPG, PNG e H.264);
- **Free Type**, uno strumento adoperato per dare una struttura rettangolare (*raster*) ai caratteri in bitmap e fornire supporto per altre operazioni legate ai font.
- **LibWebCore (WebKit)**, un framework che fornisce le basi per costruire un web browser compatibile con WebKit.

Come già detto, il framework applicativo è quella parte di architettura che contiene tutti componenti necessari per la realizzazione delle applicazioni. Fra i principali tool vengono annoverati:

- View System: permette alle applicazioni di accedere ai dati di altre applicazioni o di condividerne i propri.
- Resource Manager: garantisce l'accesso a risorse che non rappresentano codice (stringhe, grafici e file di layout).
- Notification Manager: permette a tutte le applicazioni di visualizzare alert personalizzati nella barra di stato.
- Activity Manager: gestisce il ciclo di vita delle applicazioni e fornisce un meccanismo elementare per la navigazione tra le varie schermate.
- Telephony Manager: fornisce le funzionalità di base per la telefonia.
- Location Manager: consente alle applicazioni di ottenere aggiornamenti periodici quali la posizione geografica del dispositivo; Attiva un evento specificato da un'applicazione quando il dispositivo entra in prossimità di una data posizione geografica.
- Window Manager: crea superfici di disegno per l'applicazione ed è responsabile dell'organizzazione dello schermo e della visualizzazione dei diversi livelli di cui si compone un'applicazione.