

Localization

Il sistema di localizzazione nelle Cyber-Physical-Systems (CPS) o Wireless Sensor Networks (WSN) è l'abilità di determinare la posizione dei componenti, sensori e eventi. Esso consiste in due blocchi principali: Nodes e localization algorithms.

Nodes

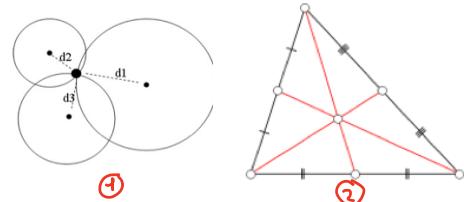
I nodi possono avere tre differenti stati:

- 1) Beacon (anchors): Sono nodi che conoscono la loro posizione tramite riabbattimento manuale
 - o GPS reading
- 2) Unknown Nodes (target Nodes): Sono nodi che non hanno nessuna info rispetto alla loro posizione.
- 3) Settled Nodes: Unknown nodes che riescono a determinare/stimare la loro posizione.

I nodi prima del localization algorithm possono essere in beacon o unknown status.

Sia i beacon che i settled nodes sono utili agli unknown nodes per stimare la loro posizione. Solitamente viene validata la distanza usando:
1) Trilateration (basata sul range)

- 2) Centroid (Range free)



Localization Algorithms

Localization algorithms hanno come obiettivo la localizzazione degli unknown nodes.

I vari metodi di localizzazione dipendono dall'obiettivo della CPS application e le sue tecnologie.

GPS: È la soluzione intuitiva per determinare la posizione accuratamente

Problema: Non sempre è il miglior metodo dovuto ai vincoli di costo e energia.

Non funziona in buon o quando i satelliti sono sconnessi.

GPS-Free techniques: Usano le capacità sensoriali e di comunicazione dei CPS components

→ RSS (Received Signal Strength) per inferire la distanza/angoli e poi stimare la posizione tramite

alcuni calcoli geometrici.

- Stimare le distanze usando la differenza tra le proprietà di propagazione di onde elettromagnetiche (es: Reception time tra le onde acustiche e radio)
- Range-free localization determina la posizione degli unknown nodes basandosi sulle info di proximità / connettività.

Tipi di localization systems

1) Remote positioning: la posizione di un nodo è calcolata ad una base centrale.

→ vari segnali radio dei mobile nodes vengono collettiati dagli anchor nodes e inviati alla base centrale, la quale stima la posizione del mobile node basandosi sui segnali inviati dagli anchors.

→ Centralized: High accuracy, non scalabile, multi-hop

2) Self-positioning: la posizione di un nodo è calcolata da esso basandosi sui measured signals ricevuti dagli anchor nodes.

→ localized: Più semplice, più energia richiesta dovuta alle più iterazioni

3) Indirect remote positioning: Il nodo calcola la sua posizione e la invia alla base station, tramite un canale wireless.

4) Indirect self-positioning: la base station calcola la posizione di un nodo e gliela invia, tramite un canale wireless.

Coordinate Systems

1) Absolute: la localizzazione è descritta in relazione agli special nodes, solitamente anchor nodes (sono necessari quattro nodi per un sistema di coordinate 3D)

2) Relative: la localizzazione è descritta basandosi sugli altri nodi non facendo riferimento agli absolute anchors (reference nodes non sono assoluti)

- 3) Physical: la localizzazione è descritta come un punto in un sistema di coordinate 2/3D
- 4) Symbolic: la localizzazione è descritta usando logical positions' information
(cell number, office, street name ...)

Performance Metrics

Accuracy: mean distance error, usually the average Euclidean distance between the estimated location and the true location.

Precision: reveals the variation in algorithm's performance over many trials. Usually, the cumulative probability functions (CDF) of the distance error.

Complexity: can be attributed to hardware, software, and operation factors. Usually, location rate is an indirect indicator for complexity.

Robustness: the system's capability of working when some signals are not available (e.g., when some of the RSS value or angle character are never seen before).

Scalability: the positioning performance degrades when the distance between the transmitter and receiver increases. A location system may need to scale on two axes: geography and density.

Cost infrastructure, maintenance, additional nodes...

Communication Paradigms

- 1) **Non-cooperative:**
 - the communication is restricted between unknown nodes and anchors (no communication between nodes with unknown locations)
 - high density of anchors or long-range anchor transmissions are needed to ensure that each unknown node is within the communication range of at least three anchors
- 2) **Cooperative:**
 - in addition, it allows communication between unknown nodes
 - anchor density is alleviated, but more processing is required
- 3) **Opportunistic:**
 - exploits interactions between existing nodes and other nodes (which may be of heterogeneous nature) that occasionally pass in their proximity
 - requires efficient node discovery and data exchange

Range-based methods (costose ma accurate)

Sono metodi di localizzazione basati su due fasi: Ranging Phase e Estimation Phase.

Ranging Phase

Ogni nodo stima la distanza o l'angolo dagli anchor vicini.

Metodi per misurare la distanza:

- 1) Potenza del segnale wireless: Usa Received Signal Strength index (RSSI)
- 2) Time of Arrival (ToA, TDoA): Usa la radio, acoustic, ultrasuoni
- 3) Angle of Arrival (AoA): Usa una directive antenna o un array di antenne

Estimation Phase

Fase in cui viene calcolata la distanza basandosi sui dati della ranging phase.

- 1) Trilateration: Determina la distanza misurando le dist. dai reference points.
- 2) Triangulation: Determina la posizione misurando gli angoli tra i reference points.
- 3) Multi-lateration: Considera tutti i beacon disponibili.

Received Signal Strength (RSS)

I target nodes stimano la distanza dagli anchors usando l'attenuazione dell'emitted signal strength.

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2}$$

In Free space: L'RSS varia in modo inversamente proporzionale al quadrato della distanza.

Not Free space (site specific): Nei casi indoor il segnale è meno potente e più esposto a errori.

I parametri che vengono usati in questi casi sono site-specific.

Viene derivato un modello empirico

ottenendo un Least square fit \leftarrow

per ogni livello di potenza.

$P_{RSSI} = \frac{X}{r^n}$ \rightarrow constants site-specific

\rightarrow distance of the radius of the circle

Time of Arrival (ToA)

ToA consiste nel calcolare il tempo di propagazione one-way dei segnali radio tra due nodi sincronizzati. Questo tempo è direttamente proporzionale alla distanza tra i due nodi.

In un posizionamento 2D, ToA measurements devono essere fatti con almeno 3 anchors.

Problema: ToA usando gli RF signals non è appropriato per le WSN dovuto alle porte distante e alla sincronizzazione richiesta.

Time Difference of Arrival (TDoA)

TDoA usa gli RF signals e gli ultrasuoni.

1) Calcola la differenza tra RF e gli ultrasuoni ($T_2 - T_1$)

2) Stima la distanza usando la velocità del suono (s)

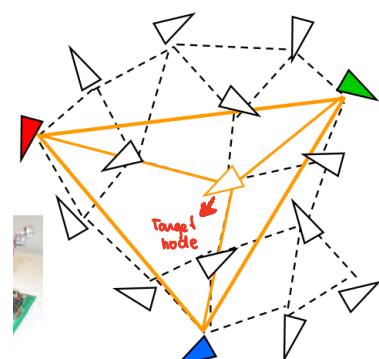
Problemi:

- Richiede molta energia
- Richiede hardware aggiuntivo

Angle of Arrival (AoA)

Usa una antenna direzionale o un array di antenne.

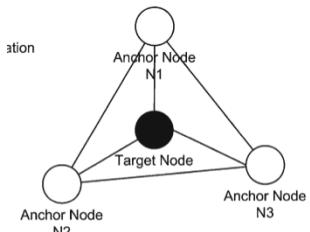
Stima gli angoli relativi tra nodi vicini.



La posizione viene derivata dall'intersezione di diverse paia di angle direction lines.

Per fare la misurazione in 3D necessita 3 misure e 2 per il 2D.

Non richiede un meccanismo di sincronizzazione.



Limitazioni dei metodi range-based

Molti range-based localization schemes sono basati su assunzioni che spesso non tengono nella pratica.

- circular radio range
- symmetric radio connectivity
- additional hardware (e.g., ultrasonic)
- lack of obstructions
- clear line-of-sight
- no multipath and flat terrain

Range-Free localization (Poco costose ma poco accurate)

Sono metodi che non fanno affidamento sulla stima delle distanza/angolo.

Utilizzano le info di prossimità o connettività.

I nodi non provano mai a stimare la loro absolute distance dagli anchors.

Vantaggi: Hardware poco costoso e bassa computational power richiesta.

Svantaggi: Meno accuratezza rispetto ai metodi range-based.

Area-based Approaches: Sono approcci che basandosi sulla radio connectivity tra il target e gli anchors, stimano la posizione del target come un punto particolare nel poligono formato dai vicini.

Multi-hop localization approaches: Sono approcci che consistono nel flooding della rete e successiva identificazione delle posizioni in base alla distanza dagli anchors in termini di hops.

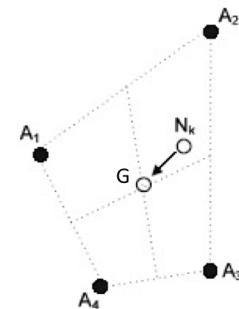
Centroid Method (Area-based Approach)

Attraverso il beacon in ascolto, il target N_k calcola la sua posizione come centroid point, definito come il geometric center di un insieme di anchors.

- Vantaggi:**
- Semplice da implementare
 - Bassa overhead (uso di pochi beacon, computazione semplice)

Svantaggi:

- Bassa accuracy
- Necessita overlapped anchors per la corretta stima



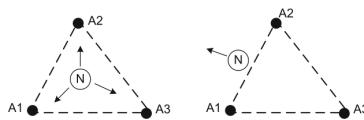
$$(X_G, Y_G) = \left(\frac{\sum_{i=1}^n (X_i)}{n}, \frac{\sum_{i=1}^n (Y_i)}{n} \right)$$

APiT (Area-based Approach)

L'idea consiste nel dividere l'environment in regioni triangolari e successivamente testare se il target risiede in un determinato triangolo o meno per nestigare l'area delle possibili posizioni del target.

È basato sul Point-In-Triangulation Test (PiT), il quale determina se un nodo N con una posizione sconosciuta è all'interno o fuori del triangolo formato dagli anchors A_1, A_2, A_3 .

Perfect PiT



Proposizione 1: Se N è all'interno del triangolo, quando N si muove in una qualsiasi direzione, la nuova posizione sarà più vicina/lontana da almeno un anchor.

Proposizione 2: Se N è al di fuori del triangolo, esiste una direzione t.c., quando N si muove lungo essa, la posizione di N diventa più lontana/vicina rispetto a tutti e tre gli anchors.

Il PiT test come sopra descritto non è applicabile nella pratica:

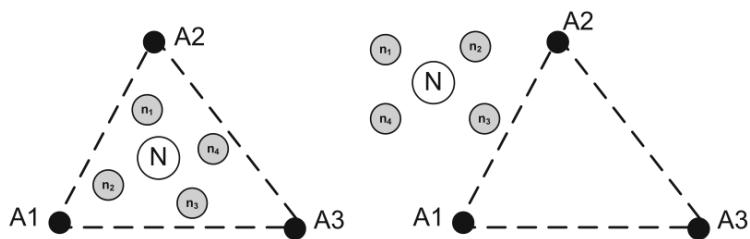
- I nodi solitamente non si muovono e non hanno l'abilità di riconoscere la direzione senza muoversi.

- Non è possibile fare un test esauritivo che ricopre tutte le possibili dimensioni dove un target può muoversi.

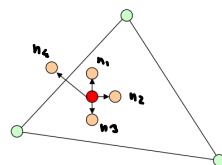
Soluzione: Usare un' approssimazione , cioè usare le info del vicinato , scambiate via beaconing , per emulare il node movement nel Perfect PIT.

Approximate PIT

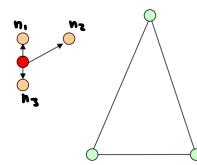
- 1) Il target node chiede ai suoi vicini la loro distanza da 3 corner anchors
- 2) Il target compara la sua distanza da questi 3 corner anchors rispetto a quelle dei vicini.
- 3) Se nessun vicino è più lontano/vicino ai 3 anchors contemporaneamente , il target assume di esserne all'interno del triangolo , altrimenti il target assume di essere al di fuori.



L' APIt test è in esecuzione nei seguenti scenari :



In-to-out error



Out-to-in error

Il nodo fuori dal triangolo (n_4) è più distante , rispetto al target node , da tutti e tre gli anchors .

L' APIt quindi: darebbe un risultato sbagliato , dicendo che il target è fuori .

Nessun nodo è più lontano/vicino da tutti e tre i nodi

L' APIt quindi: darebbe un risultato sbagliato , dicendo che il target è dentro .

Relative distance estimation: Più è lontano il nodo dall'anchor, più debole è la potenza del segnale ricevuto.

Apit Aggregation: Rappresenta la massima area nella quale un nodo è probabile risieda usando un grid SCAN algorithm.

- Per la inside decision le grid regions sono incrementate
 - Per la outside decision le grid regions sono decrementate

Algorithm:

Receive location beacons (X_i, Y_i) from N anchors;
 $InsideSet = \emptyset$;

```

InsideSet = {};
for each triangle  $T_i \in \binom{N}{3}$  triangles do
    if Point-In-Triangle-Test ( $T_i$ ) == TRUE then
        InsideSet = InsideSet  $\cup T_i$ ;
    end if
end for
Estimated Position = CenterOfGravity(  $\bigcap T_i \in InsideSet$ )

```

Estimated Position = CenterOfGravity($\bigcap T_i \in InsideSet$);] Intervento di tutti i triangoli che hanno dato riscontro positivo all' API test

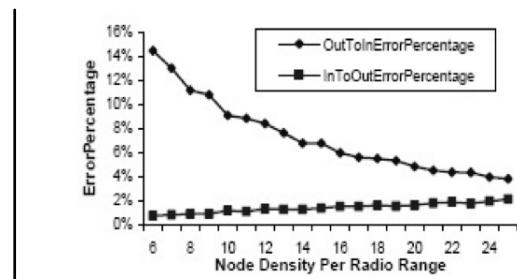
- Vantaggi: • Piccolo overhead
• Più risultati accurati del centroid method

Drawbacks: Ci sono problemi nel determinare un nodo situato fuori da tutti gli anchor triangles (undetermined target)

Experimental results:

I risultati sperimentali mostrano che la percentuale degli errori è minore come la densità dei nodi aumenta

ne di tutti i triangoli che hanno dato riscontro positivo all'APit test.



DV-Hop (Multi-hop localization approaches)

- 1) Flooding the network t.e ogni anchor trasmetta indipendentemente un beacon, che contiene la sua location e un hop-counter field settato inizialmente a uno e aumentato ad ogni nuovo hop.
 - 2) Ogni target identifica il path più corto verso ogni anchor node e prova a stimare la distanza da esso

L'idea è di calcolare il numero di hops tra ogni due anchor (A_i, A_j) e stimare la avg 1-hop distance dividendo la somma delle distanze fisiche per la somma delle distanze logiche.

Node update phase:

- Quando un target N riceve un beacon da un anchor, mantiene il record (x, y, h) per ogni anchor A.

(x, y) = Posizione dell'anchor

h = Numero di hops da N all'anchor A.

- N aumenta h e inoltra il beacon ai suoi vicini (all'interno del radio range).

Alla fine di questo step, i target sanno la posizione degli anchor nodes e il numero di hop per raggiungerli.

+Hop distance estimation phase

- Quando un anchor node riceve la posizione e l'hop count verso gli altri anchors, calcola la estimated avg +hop distance riferita come connection factor (c_i).

$$c_i = \frac{\sum (\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2})}{\sum(h_i)} \rightarrow \text{Somma della distanza verso gli altri anchors, diviso la somma degli hops verso gli altri anchors}$$

- L'anchor floods the network con le estimated +hop distance. Un nodo che riceve un connection factor, lo inoltra e smette di inoltrare i successivi connection factors.

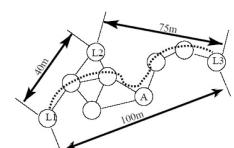
Target Node localization:

Ogni target usa il connection factor ricevuto dal closest anchor come estimated +hop distance. Successivamente moltiplica la +hop distance per il numero di hop verso gli altri anchors per stimare la sua distanza fisica verso loro.

- Dopo aver ottenuto la distanza stimata verso almeno 3 anchors, un target può usare la trilaterazione per approssimare la sua posizione.

Vantaggi: Semplice

- Svantaggi:
- Estimation non dipende dal numero di anchors che un nodo può ascoltare
 - Large overhead.
 - Funziona solo per network uniformi in tutte le orientations



L_1 computes the correction $\frac{100+40}{6+2} = 17.5$

L_2 computes a correction of $\frac{75+25}{2+5} = 16.42$

L_3 a correction of $\frac{75+100}{6+5} = 15.90$

A gets its correction from L_2

Distance to $L_1 = 3 \times 16.42$, to $L_2 = 2 \times 16.42$, to $L_3 = 3 \times 16.42$

Publish - subscribe pattern

È un messaging pattern dove i publisher inviano i messaggi, che vengono divisi in classi, e i subscribers ricevono solo i messaggi dai canali a cui sono iscritti.

Filtering

Per implementare questo concetto, abbiamo bisogno di filtrare i messaggi.

1) Topic-based system: i messaggi sono pubblicati in "topics" o named logical channels.

↓
i mess sono filtrati in i subscribers riceveranno tutti i messaggi pubblicati nei topics a cui base al topic
sono iscritti.

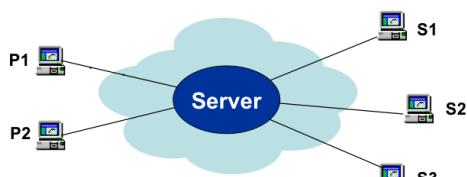
Il publisher è responsabile di definire i topics ai quali un subscriber si può iscrivere.

2) Content-based system: i messaggi vengono consegnati al subscriber se gli attributi o il

↓
contento di questi messaggi corrispondono ai vincoli definiti dai subscribers.

Il subscriber è responsabile di classificare i messaggi.

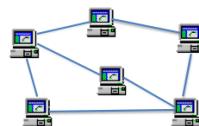
Client - Server solution



+ Simple

- A lot of traffic and computation in the server }
 ↳ the server distributes mess. to subscribers
- Not scalable
- Single Point of Failure

P2P solution



Distributed system architecture:

Nodes are symmetric in function (ogni nodo può essere un P o un S)

No centralized control

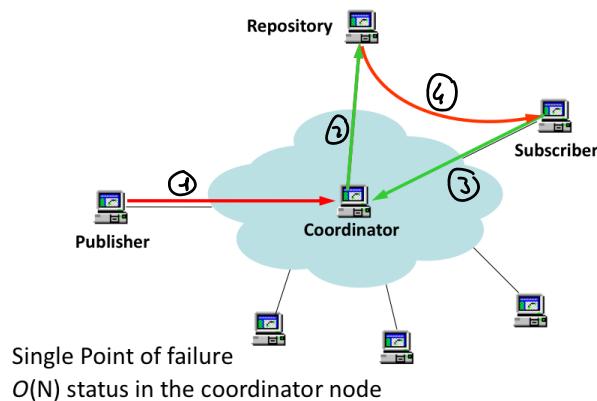
Large number of heterogeneous and unreliable nodes

↓
anche se i nodi sono unreliable, ci sono molte repliche
e di conseguenza il sistema in sé può essere considerato reliable

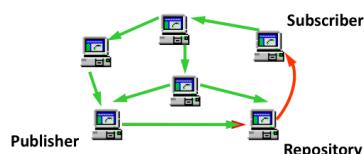
P2P Lookup problem

In un approccio P2P dobbiamo capire quale è il giusto repository node che contiene la risorsa cercata. Le risorse sono associate a una chiave. \downarrow Nodo che contiene le risorse

- 1) Centralized Approach: Ye coordinazione si occupa di assegnare una risorsa a un central repository e di permettere ai subscriber di recuperarla.



- 2) Flooding Approach: Non c'è un coordinatore centrale, tutti i nodi possono fare sia da publisher, che da subscriber, che da repository. Il problema sta nel capire quale nodo ha la risorsa cercata.



$O(N)$ messages per lookup

Distributed Hash Table (DHT)

La DHT fornisce un lookup service simile alle hash table:

- 1) ($key, value$) pairs vengono salvate in una DHT
 - 2) Ogni nodo partecipante può efficientemente recuperare il valore associato a una data chiave
- Le chiavi identificano univocamente un valore.

la responsabilità del mantenere il mapping tra chiavi e valori è distribuita tra i nodi.

Le più grande vantaggio delle DHT è che i nodi possono essere aggiunti / rimossi con il minimo di lavoro nella ridistribuzione delle chiavi.

DHT structure

La DHT structure è composta da:

- 1) Un Keyspace, cioè l'insieme di bit che compongono le strings (ad esempio 160)
- 2) Un Keyspace partitioning scheme, il quale divide il Keyspace su più nodi
- 3) Un overlay network, il quale connette i nodi permettendogli di trovare il proprietario di ogni chiave nel Keyspace

DHT operations

I dati sono identificati univocamente da una chiave e le chiavi sono distribuite tra i nodi.

DHT permette di svolgere le operazioni di:

- 1) insert (key, data) = Inserire un nuovo dato con la chiave associata
- 2) lookup (key) = Trovare il dato associato a una chiave

Un uso tipico di queste operazioni è il seguente:

- 1) Viene generata la chiave k a 160 bit usando la SHA-1 sul filename
- 2) Viene mandato un messaggio $\text{insert}(k, \text{data})$ ad un nodo qualsiasi della DHT
- 3) Il messaggio viene inviato da nodo a nodo fin quando non raggiunge il nodo responsabile per la chiave k come specificato dal Keyspace partitioning scheme. Il nodo salva key e data
- 4) Un altro qualsiasi client può richiedere il contenuto del file facendo l'hashing sul filename per produrre k e chiedere ad un qualsiasi nodo della DHT di trovare i data associati a k tramite un $\text{lookup}(k)$ message.
- 5) Il messaggio verrà inviato fino al nodo responsabile che risponderà con i data salvati.

DHT Properties

- **Autonomy and decentralization:** the nodes collectively form the system without any central coordination
- **Scalability:** the system should function efficiently even with thousands or millions of nodes
- **Fault tolerance:** the system should be reliable (in some sense) even with nodes continuously joining, leaving, and failing
- **Load Balancing:** the hash function should evenly distribute keys to nodes
- **Anonymity:** can be achieved by special routing overlay networks that hide the physical location of each node from other participants

OSS: DHT message lookup è $O(\log(N))$ e DHT richiede la costruzione e la manutenzione delle routing tables.

PASTRY

Pastry è una overlay network P2P per le DHT.

le key-value pairs sono salvate in una rete P2P ridondante di host connessi a internet.

Grazie al sua ridondanza e decentralizzazione, non c'è un singolo point of failure e i nodi possono uscire dalla rete in un qualsiasi momento.

PASTRY Design

La topologia del DHT's key-space è circolare.

I node IDs e chiavi sono unsigned integer da 128-bit (in base 2^b), i quali rappresentano la posizione nello spazio circolare

Node IDs: Vengono scelti randomicamente e uniformemente, in modo che peers adiacenti nel node ID sono geograficamente distanti.

Sono univoci e sono assegnati quando i nodi entrano nel sistema basandosi su un **secure hash** del IP address del nodo

Keys: Vengono salvate nel nodo con il node ID numericamente più vicino alla chiave.

Pastry Routing

Dato un msg e una chiave, esso viene inviato al nodo con il node ID più vicino alla chiave.

Assumendo un network di N nodi, questo può essere fatto in meno di $\lceil \log_{2^b} N \rceil$ passi: in media.

Anche se i nodi falliscono, l'eventual delivery è garantita fin quando $\frac{1}{2}$ o più nodi con node IDs adiacenti non falliscono simultaneamente.

Parametro può ed intero, tipicamente = 16

Routing Table

NodeIDs e chiavi sono una sequenza di digit in base 2^b .

La routing table di un nodo è organizzata in $\lceil \log_2 N \rceil$ righe e 2^b colonne.

Le entries nella riga n fanno riferimento al nodo il quale nodeID corrisponde al nodeID del nodo attuale nei primi n digits, ma il cui $(n+1)$ -esimo digit ha uno dei 2^{b-1} possibili valori tranne il $(n+1)$ -esimo digit presente nel nodeID del nodo attuale. Ogni entries nella routing table fa riferimento a uno dei potenziali molti nodi: qualche nodeID ha il prefisso appropriato. Tra questi nodi, il più vicino al nodo attuale (secondo una metrica di prossimità) viene scelto.

Routing Table of a Pastry node with:

nodeID = 65a1x

$b = 4$

- digits are in base 16
- x represents an arbitrary suffix
- the IP address associated with each entry is not shown

0	1	2	3	4	5	7	8	9	a	b	c	d	e	f
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
0	1	2	3	4	5	6	7	8	9	b	c	d	e	f
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
a	a	a	a	a	a	a	a	a	a	a	a	a	a	a
0	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

leaf Set

In aggiunta alle routing table, ogni nodo mantiene l'IP address dei nodi nel suo leaf set, cioè l'insieme dei:

- $\ell/2$ nodi numericamente più vicini al nodeID con il nodeID più grande.
- $\ell/2$ nodi numericamente più vicini al nodeID con il nodeID più piccolo.

Leaf set	SMALLER	LARGER
10233033	10233021	10233120
10233001	10233000	10233230

Nodes numerically closer to the present Node

Routing table	1	-2-2301203	-3-1203203
-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	2	10-3-23302	
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
	2		

Prefix-based routing entries:
common prefix with
10233102-next digit-rest of
nodeID

Neighborhood set	13021022	10200230	11301233	31301233
	02212102	22301203	31203203	33213321

Nodes "physically" closest to the present node

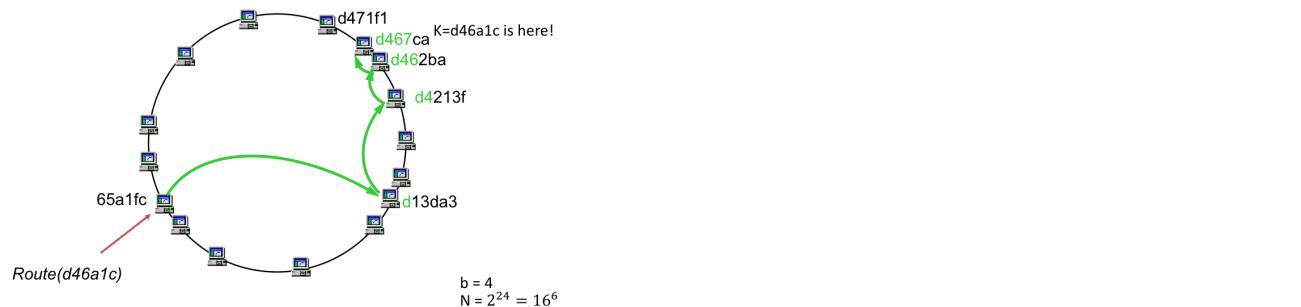
È un valore scalare che riflette la distanza tra ogni paio di nodi, ad esempio il round trip time.

È assunto che esiste una funzione che permette a ogni nodo di determinare la distanza tra se stesso e un altro nodo.

nodeId=10233102, $b = 2, l = 8$

Routing steps

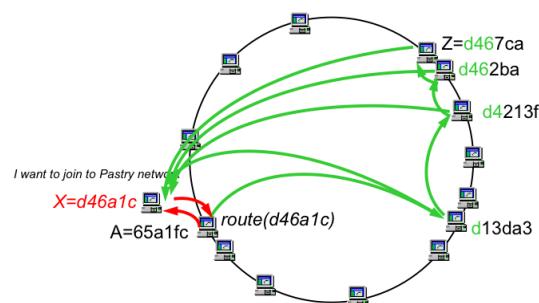
- 1) Ynolha il mex al nodo il quale nodeID condivide con la chiave un prefisso che è almeno un digit più lungo del prefisso che la chiave condivide con il nodeID corrente.
- 2) Se nessun nodo viene trovato nella routing table, verifica il leaf set.
Il messaggio viene inviato al nodo il quale nodeID condivide un prefisso con la chiave lungo come il nodo corrente, ma è numericamente più vicino alla chiave rispetto al nodeID corrente
- 3) Se il nodo non esiste in nessuno dei due casi, la chiave viene salvata nel nodo corrente (a meno che gli l/2 nodi adiacenti nel leaf set hanno fallito contemporaneamente)



Pastry node arrival

Quando un nodo vuole entrare nella rete con il nodeID = X

- 1) Contatta un Pastry node A conosciuto e vicino e chiede di instaurare un join mex all'interno della rete con X come chiave
- 2) Il mex viene instaurato verso il nodo z che ha il nodeID numericamente vicino a X
- 3) X ottiene il leaf set da z, e lo i-th riga della routing table dall'i-th node incontrato lungo la via da A a z
- 4) X notifica i nodi per aggiornare i loro stati



Pastry node departure or failure

g nodi vicini nel nodeId space (che sono l'uno nel leaf set dell'altro) periodicamente scambiano dei keep-alive messages.

Se un nodo non risponde per un periodo T, viene presupposto che è failed.

Tutti i membri che fanno parte del leaf set del nodo fallito vengono notificati e aggiornano i loro leaf sets. Essendo che i leaf sets dei nodi con nodeIDs adiacenti si sovrappongono, l'update è banale.

Un recovering node contatta i nodi nel suo ultimo leaf set conosciuto, ottiene il current leaf set, aggiorna il suo leaf set e notifica i membri del suo nuovo leaf set della sua presenza. Le routing table entries che fanno riferimento a failed nodes vengono riparate rapidamente.

Pastry short route property

The total distance, in terms of the proximity metric, that messages travel along Pastry routes:

1. each entry in the node routing tables is chosen to refer to the nearest node, according to the proximity metric, with the appropriate nodeId prefix
2. as a result, in each step a message is routed to the nearest node with a longer prefix match

Pastry route convergence property

The distance traveled by two messages sent to the same key before their routes converge

- Simulations show that, given our network topology model, the average distance traveled by each of the two messages before their routes converge is approximately equal to the distance between their respective source nodes

Pastry API:

`nodeId = pastryInit(Credentials)`

- a. causes the local node to join an existing Pastry network (or start a new one) and initialize all relevant state
- b. returns the local node's nodeId

The credentials are provided by the application and contain information needed to authenticate the local node and to securely join the Pastry network

`route(msg, key)`

- causes Pastry to route the given message to the node with nodeId numerically closest to key, among all live Pastry nodes

`send(msg, IP-addr)`

- causes Pastry to send the given message to the node with the specified IP address, if that node is alive. The message is received by that node through the deliver method

le appl. in top a Pastry devono esportare le sequenze operazioni:

- 1) `deliver(msg, key)`
 - called by Pastry when a message is received and the local node's nodeId is numerically closest to key among all live nodes, or when a message is received that was transmitted via `send`, using the IP address of the local node
- 2) `forward(msg, key, nextId)`
 - called by Pastry just before a message is forwarded to the node with nodeId = nextId. The application may change the contents of the message or the value of nextId. Setting the nextId to NULL will terminate the message at the local node
- 3) `newLeafs(leafSet)`
 - called by Pastry whenever there is a change in the leaf set. This provides the application with an opportunity to adjust application-specific invariants based on the leaf set

SCRIBE

Un publish-subscribe system che usa Pastry.

Ogni scribe node può creare un gruppo. Gli altri nodi possono joinare il gruppo, o multicastare i messaggi a tutti i membri del gruppo.

La rete di scribe è una P2P network di Pastry nodes ed è completamente decentralizzata, tutte le decisioni sono basate sulla info locali e ogni nodo ha capacità identiche.

Scribe Publish - Subscribe implementation

Ogni Topic (gruppo) ha un topicID. Il TopicID è l'hash del nome del topic in formato testuale + il creation name.

L'hash è calcolato tramite una hash function collision resistant (es: SHA-1), la quale assicura una distribuzione uniforme dei topicIDs.

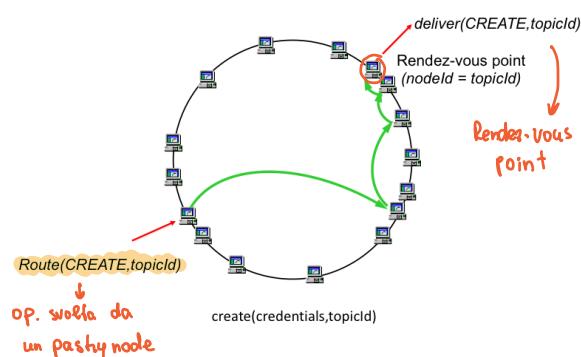
Essendo che i pastry nodeID sono anch'essi uniformemente distribuiti, questo assicura una distribuzione equa dei topics tra i vari pastry nodes.

Rendezvous point = Scribe node con un nodeID numericamente vicino al topicID.

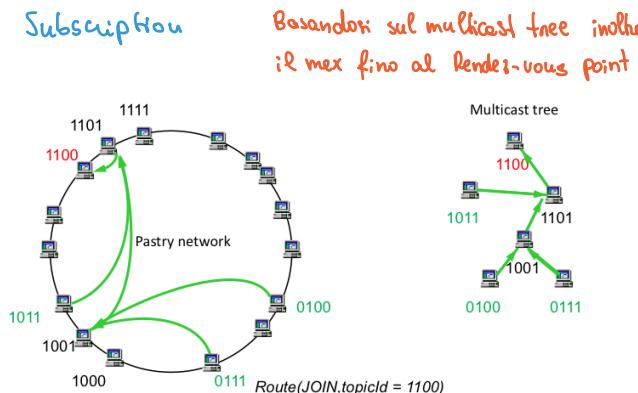
Esso è la radice del multicast tree creato per un certo topic (gruppo)

Create a Topic

- 1) Viene creato un multicast tree, con radice il rendezvous point, per disseminare i multicast messages nel gruppo.
- 2) Il multicast tree viene creato usando uno schema simile al reverse path forwarding.
- 3) L'albero è formato unendo i percorsi da ciascun membro al rendez-vous point.



Subscription



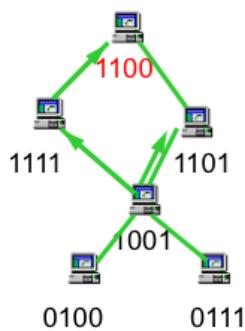
Message Dissemination

- Route through the Pastry network using the topicId as the destination

- Dissemination along the multicast tree starting from the rendez-vous

per inoltrare a tutti i membri del gruppo

Repairing the multicast tree



Scribe API

- create(credentials, groupId)**
creates a group with groupId. Throughout, the credentials are used for access control
- join(credentials, groupId, messageHandler)**
causes the local node to join the group with groupId. All subsequently received multicast messages for that group are passed to the specified message handler
- leave(credentials, groupId)**
causes the local node to leave the group with groupId
- multicast(credentials, groupId, message)**
causes the message to be multicast within the group with groupId

Human as Sensor (Has)

La grande disponibilità dei contenuti sui social network suggerisce che la più grande sensor network possono essere gli umani.

Questa rete sarebbe composta da:

- g sensori : le risonse umane
- g dati : le osservazioni che vengono fatte sul mondo fisico da parte degli umani (claims)

Problema: L'affidabilità di un human sensor è generalmente sconosciuta a priori (l'osservazione è vera o falsa?) e la provenienza delle osservazioni può essere incerta (gli individui potrebbero reportare observations, non fatte da loro, come se fossero loro)

→ Dati questi due problemi, bisogna determinare se un claim è corretto.

Has architecture

Per fare la ricostruzione delle verità dai dati riportati dagli human sensor, dobbiamo:

1. Collezione i dati dalla sensor network
2. Strutturare i dati per l'analisi
3. Capire come le risorse sono relazionate
4. Usare questa info collettiva per stimare la probabilità delle connettività di un'osservazione individuale.

1. Data Collection

In principio, i dati possono essere raccolti da una qualsiasi media platform.

No: supponiamo che vengano raccolti da Twitter, tramite un API che permette il matching data una keyword e, optionalmente, un area geografica.

2. Data Structuring

1) Calcolo della distance function

Viene calcolata una distance function $distance(t_1, t_2)$ per ogni paio di repeated observations.

Questa funzione ritorna la loro similarità.

Nel caso dei tweets:

- a) le observations sono i tweet, i quali vengono divisi in tokens.
- b) A e B sono i vettori delle occorrenze dei token all'interno dei due tweet
- c) Viene usata la cosine similarity function che ritorna la similarità in base al numero di matching tokens $similarity = \cos \theta = \frac{A \cdot B}{\|A\| \|B\|}$ dot product vector magnitude

2) Costruzione delle "claims"

- a) Viene costituito un grafo dove i vertici sono le osservazioni e i link rappresentano la loro similarità.
- b) Viene fatto il cluster del grafo, dove osservazioni simili vengono raggruppate

c) Ogni cluster viene chiamato "claim", definito come un settore di info che diverse sorgenti hanno riportato

3) Costruzione del Source-claim graph (sc)

Viene quindi costruito l' sc graph, nel quale :

- Ogni sorgente, s_i , è connessa a tutti i claim nei quali ha contribuito
- Ogni claim, c_j , è connesso a tutte le sources che lo "hanno esposto"
 $s_i \cdot c_j = 1 \Leftrightarrow s_i \text{ dichiara } c_j$

Ogni claim è True se è consistente con la verità nel mondo fisico. Altrimenti False!

3. Sources Relationship

Le sources possono sia aver riportato le loro obs, sia quelle di qualcun altro (uncertain provenance)

Assumiamo l'esistenza di un social information dissemination graph (SD), il quale stima come l'info potrebbe essere propagata da una persona a un'altra:

a) Epidemic Cascade Network (EC): Ogni obs distinta è modellata come una cascata e il tempo di contagio di una source descrive quando la source ha mentionato l'obs.

b) Follower - Follower Network (FF): Esiste un link diretto (s_i, s_k) nel social graph dalla source s_i alla source s_k , se s_k segue s_i

c) Re-Tweeting Network (RT): Esiste un link diretto (s_i, s_k) nel social graph se la source s_k retweets alcuni tweets dalla source s_i

d) Combined Network (RT+FF): Esiste un link diretto (s_i, s_k) quando sia s_k segue s_i oppure s_k retweets quello detto da s_i

4. Estimation Problem

Bisogna stimare la connettività dei claims, cioè per ogni claim, c_j , dobbiamo determinare se è vero o falso.

y tweets vengono analizzati tramite una sliding window, cioè consideriamo come N il numero totale di claims calcolati dai tweets ricevuti nell'ultima window.

Come determinare la connettività di un claim?

1) Voting: y claim con più supporto sono i più credibili. } claim che sono stati citati da più sorgenti

$$\begin{array}{l} \rightarrow \forall c_j, 1 \leq j \leq N, \text{ conta tutti gli } s_i \text{ dove } s_i c_j = 1 \\ \rightarrow \text{Problema:} \end{array}$$

4) Sources differenti hanno diversi gradi di reliability, quindi il voto delle sources non dovrebbero avere tutti lo stesso peso.

2) le sources potrebbero non essere indipendenti tra loro.

2) Likelihood Estimation: calcolare la prob. di connettività dei claims tenendo in considerazione una provenance e reliability sconosciuta.

\rightarrow Dat: i grafici SC e SD, qual è la likelihood che c_j è true, $\forall j$?

Formalmente stiamo calcolando:

$$\forall j, 1 \leq j \leq N : P(c_j = 1 \mid SC, SD)$$

3) Maximum Likelihood (ML) Estimation: un approccio di ottimizzazione

Siano:

- m = numero totale di sources

- le sources s_i , $1 \leq i \leq m$, vengono descritte da due parametri, i quali sono sconosciuti e devono essere calcolati:

a) odds of true positives: $a_i = P(s_i c_j = 1 \mid c_j = 1)$ } Prob. che dada un song. e un claim, il claim sia vero

b) odds of false positives: $b_i = P(s_i c_j = 1 \mid c_j = 0)$ ↓ affidabilità della sorgente

- Sia d l'expected ratio dei claims connessi nel sistema, anch'esso sconosciuto.

$$\hookrightarrow d = P(c_j = 1)$$

- Sia $\theta = [a_1 \dots a_m, b_1 \dots b_m, d]$ il vettore dei parametri sconosciuti sopra descritti

L'MLE estimator trova i valori di θ che massimizzano la prob. delle osservazioni, sc, dato il social network SD:

$$P(SC|SD, \theta)$$

Essendo che $P(SC|SD, \theta)$ dipende su quali claims sono veri e quali falsi, introduciamo un vettore z dove $z_i = 1$ se C_j è true e zero altrimenti.

Assumendo che i claims sono indipendenti, possiamo descrivere la espressione da massimizzare come segue:

$$P(SC|SD, \theta) = \sum_z P(SC, z|SD, \theta)$$

Per massimizzazione MLE possono essere usati gli algo descritti nella letteratura, data la notione di uncertain provenance dei social network che viene incorporata.

Intuizione: Dato un Social Dissemination graph (SD), se la sorgente "genitore" non fa il claim, allora la sorgente "figlio" agisce come se fossero indipendenti. Se invece la sorgente "genitore" fa il claim, allora ogni "figlio" ripete il claim con una certa probabilità.

Questo può essere fatto unendo sc e sd e dividendoli in sottoinsiemi, SC_j , i quali descrivono quali sorgenti hanno esposto C_j e quali no.

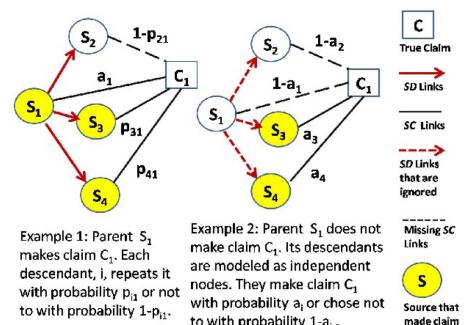
Successivamente si calcola il repeat ratio:

$$p_{ik} = \frac{\text{number of times } S_i \text{ and } S_k \text{ make same claim}}{\text{number of claims } S_k \text{ makes}}$$

Quindi l'algoritmo di massimizzazione prenderà:

Input: SC e SD

Output: Maximum likelihood estimation della source reliability e connectness.

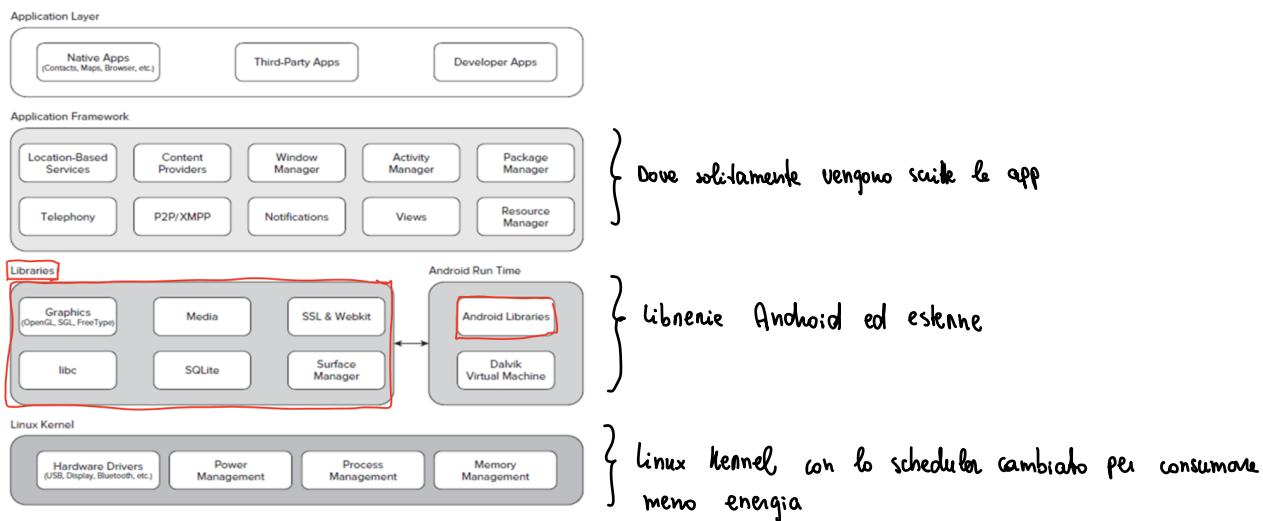


PARTE DI VECCHIO

Introduction to Android

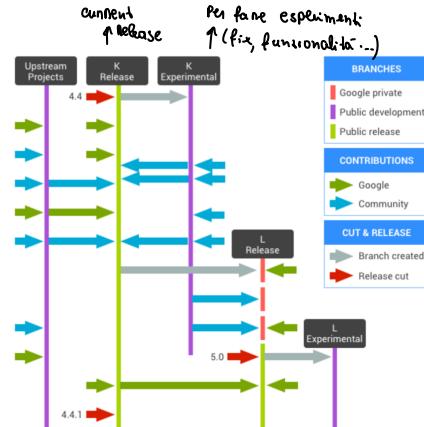
Android è il SO più popolare nell'ambito mobile, open source e mantenuto/esteso da Google

Architecture



Android Open Source Project (AOSP) : Mantiene il software

Original Equipment Manufacturer (OEM) : Azienda che porta il SW per essere eseguito sui propri dispositivi.



Android App Development

le App vengono distribuite sotto forma di APK, ogni file APK contiene:

- 1) Dex Executable
- 2) Resources (xml files, images, ...)
- 3) Native libs

Per pub. costa una fee e bisogna iscriversi

la distribuzione può avvenire tramite: Google play store, Web, email

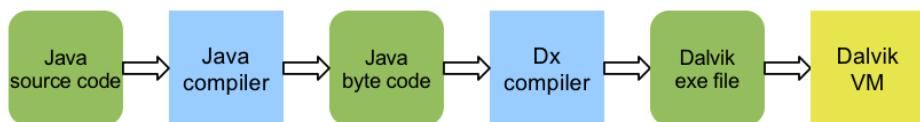
Le Applicationi possono essere scritte usando Java o Kotlin.

Java and Dalvik

Dalvik VM assicura agli sviluppatori di non preoccuparsi dell'impl. di hw particolari.

Essa esegue i file Dalvik in modo ottimizzato.

I developers creano il file .dex trasformando le classi Java compilate usando i tools del SDK (Software Developer Kit).



ART (Android Run time)

ART, a differenza di Dalvik, trasforma in linguaggio nativo l'app durante il download rispetto a Dalvik che lo trasforma sul momento della lettura del codice.

Richiede quindi:
+ tempo di installazione
- tempo di esecuzione } + veloce, - energia

PS: I file prima di essere scaricati sono Dalvik files.

Android Development Problems

Ci sono molte versioni di android con differenti caratteristiche, inoltre anche la forma dei dispositivi può variare (smartphones, tablets...).

ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.1 Jelly Bean	16	99,8%
4.2 Jelly Bean	17	99,5%
4.3 Jelly Bean	18	99,4%
4.4 KitKat	19	98,0%
5.0 Lollipop	21	97,3%
5.1 Lollipop	22	94,1%
6.0 Marshmallow	23	89,0%
7.0 Nougat	24	85,6%
7.1 Nougat	25	82,7%
8.0 Oreo	26	78,7%
8.1 Oreo	27	69,0%
9.0 Pie	28	50,8%
10.0 Q	29	24,3%
11.0 R	30	

Isolation Between Apps

Android è un S.O. linux multi-user.

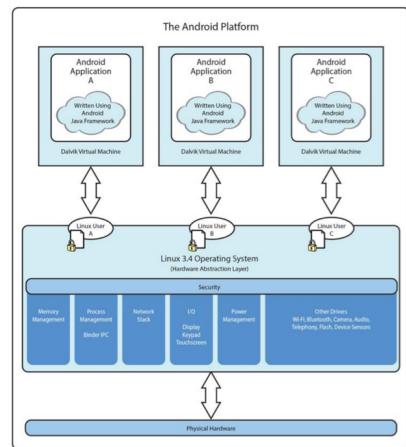
Ogni applicazione è un differente utente, con un ID assegnato dal sistema.

Access Control: Solo i processi con lo stesso ID possono accedere ai file.

Ogni processo ha la sua VM e lavora in modo isolato.



Linux Process avviato quando i componenti dell' APP devono essere eseguiti e spento quando non utilizzato per molto.



Ref: Introduction to Android Programming,
Annuzzi, Darcey & Conder

Application UI + logic

La UI di un'applicazione viene descritta in formato XML, mentre la logica in Java.

In questo modo è possibile modificare l'UI senza modificare il Java.

Le componenti principali di un'app sono:

- 1) Files Java che descrivono la logica } Vengono descritte le attività con il quale l'utente interagisce una alla volta
- 2) Files XML che descrivono l'UI
- 3) File "AndroidManifest.xml" che:
 - a) Include la lista dei componenti
 - b) Specifica l'entry point dell'esecuzione

Android GUI

Tutti i widget sono una sottoclasse di View e vengono dichiarati nel file XML.

Ogni Widget ha un ID che viene usato da Java per interagire con esso.

Android Resources

Le Android Resources sono elementi dell'APP definiti in altri file.

Salvandoli in file diversi sono più facili da aggiornare e mantenere.

Esse possono essere:

a) Usare nel Java Code (Usando la R class che fa da hamite tra Java e le risorse)

Sono identificate tramite il name attribute R.string.*my_string*
R class XML file Nome della risorsa

b) Essere referenziate da altre risorse XML.

Sono usate come attributi di alcuna XML resources `android:text="@string/my_string"`



XML file Nome della risorsa

Android DataDriven layouts

Sono layout che permettono di mostrare i dati all'utente.

Per popolare i layout con i dati viene utilizzato un adapter:

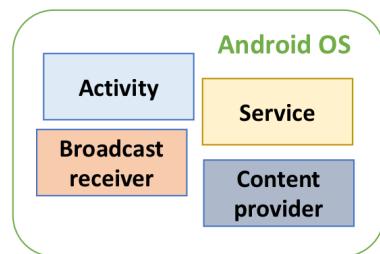
- 1) Cursor Adapter: legge da un DB
 - 2) Array Adapter: legge da una risorsa o Java array

Android Components

Le App android non hanno un main(), esse sono una collezione di componenti derivate dalle classi del S.O.

Android chiama i metodi delle componenti in base ad un approccio event-driven (guidato dagli eventi).

L' App ha 4 tipi di componenti:



- 1) Activities
 - 2) Services
 - 3) Broadcast Receivers
 - 4) Content Providers

Activities

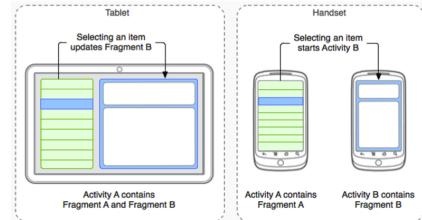
Sono i blocchi di costruzione di una UI (come le finestre in una desktop appl.)

le applicationi devono avere almeno un'attività

Un utente può interagire con una attività per volta.

le attività possono essere formate da diversi fragments

(UI blocks) che possono essere organizzati differentemente su dispositivi diversi.



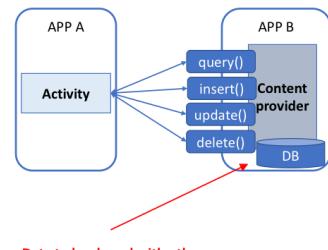
Services (sub-classes della Android's service class)

I servizi sono task che possono continuare anche in background.

le attività, solitamente, possono controllare i servizi: avviareli, metterli in pausa, prendere i dati.

Content Providers (sub-classes della Android's ContentProvider class)

le app Android possono condividere i dati come content providers, cioè mettere a disposizione dei metodi che possono essere chiamati da altre app per ottenere i dati (o anche aggiungerli).



Broadcast Receivers (sub-class della Android's BroadcastReceiver class)

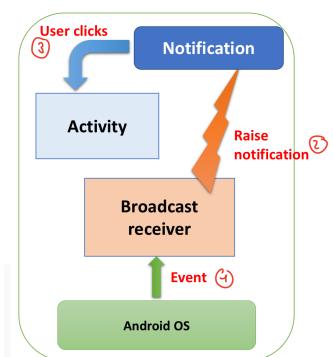
Ogni app può creare un broadcast receiver per ascoltare gli eventi broadcasted dal Android OS system o le altre applicationi.

La nostra APP può anche inviarli.

Soltanente creano una notifica per avvisare l'utente, non interagiscono con l'UI.

Interazione con l'utente:

- 1) OS broadcasts un evento
- 2) yr receiver cattura l'evento e fa una notifica.
- 3) la notifica rimane nella status bar fin quando l'utente non vuole interagire
- 4) la notifica lancia l'attività



Android Activity Lifecycle

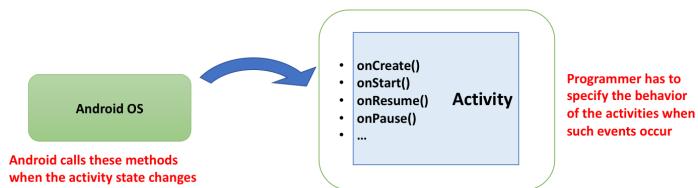
Running Application: Applicatione con la quale l'utente sta interagendo (visible)

Paused Application: Un'applicazione è in pausa se è visibile ma non in foreground.
(`onPause()`)
es: Quando avviene un pop-up.

Stopped Application: Un'applicazione è stopped se non è più né visibile né in foreground.
(`onStop()`)
es: L'utente usa un'altra app.

Può essere resumed in running state se torna visibile

Le attività hanno delle callbacks che vengono invocate in base al cambiamento di stato dell'app.

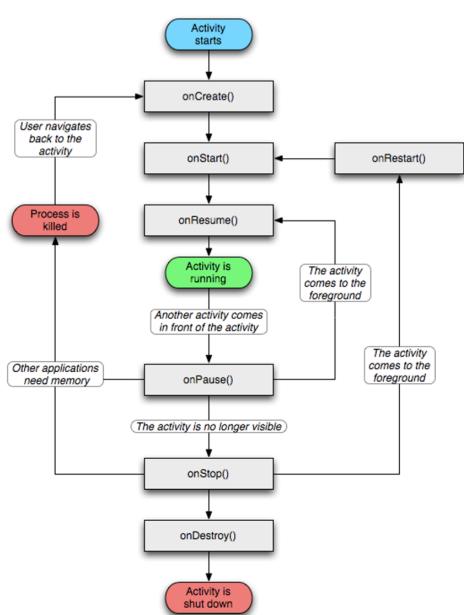


Android chiama tutti questi metodi. →

- `onCreate() -> sempre definito!`
- `onStart()`
- `onResume()`
- `onPause()`
- `onStop()`
- `onRestart()`
- `onDestroy()`

Il programmatore li definisce solo un sottoinsieme di essi:

in base a quelli che servono per la specifica attività.



OnCreate() method

Viene chiamato quando l'attività viene creata e la initializza.

- 1) Crea i widget e li mette a schermo
- 2) Ottiene la referenza sui widget (`findViewById()`)
- 3) Imposta i listening per gestire le interazioni dell'utente

OnPause () method

Viene chiamato quando l'app passa da running to pause state

Azioni tipiche:

- 1) Ferma i CPU intensive tasks, audio, video, animationi
- 2) Ferma il listening per GPS, broadcast info, ...

OnResume () method

Viene chiamato durante la transizione da paused a running state

Azioni tipiche:

- 1) Riavvia video, animationi, GPS tracking ...

OnStop () Method

Viene chiamato quando un'altra app diventa visibile e in foreground rispetto alla nostra.

Azioni Tipiche :

- 1) Salvare i progressi
- 2) Rilasciare le risorse, salvare le info

OnDestroy Method()

Viene chiamato poco prima che l'app venga distrutta.

Può essere usato per liberare risorse, come tenere i thread.

Device Rotation

Quando il device viene ruotato, la attività corrente viene killata e viene creata una nuova istanza della stessa attività, ma in landscape mode.

Portrait: Usa i file XML in res/layout

Landscape: Usa i file XML in res/layout-land/

Saving state

Il sistema, prima che l'app venga distrutta, chiama il `onSaveInstanceState()` method.

I dati vengono salvati in un "Bundle object" e dati al sistema prima che l'app venga distrutta.



Restoring App State

Quando l'attività viene ricreatata, i dati salvati vengono inviati sia al metodo `onCreate()`, che al metodo `onRestoreInstanceState()`.

Possono essere usati entrambi i metodi per fare il restore dei dati dell'app.



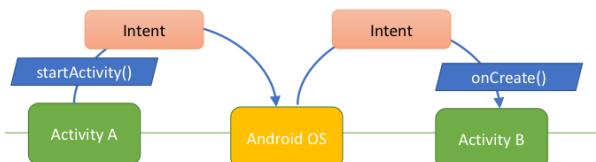
Intents

Gli intent sono messaggi usati dai componenti per richiedere azioni da un'altra app o componente.

Ci sono 3 casi principali:

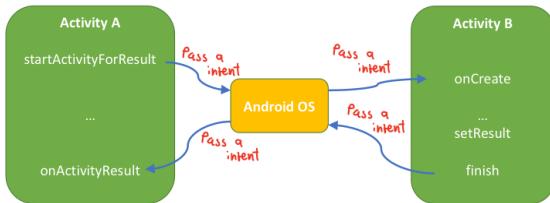
1) Attività A inizia B, nessun risultato indietro:

- Chiama il metodo `startActivity()`, passando un intent
- l'intent ha l'info su quale attività startare + dati necessari



2) Attività A inizia B, ottiene risultato indietro:

- Chiama il metodo `startActivityForResult()`, passa un intent
- Viene ricevuto un intent sulla callback `onActivityResult()` per i risultati.



3) Attivita' A starta un servizio:

- chiama il metodo `startService()`, passando un intent
- l'intent contiene le info del servizio da startare + info aggiuntive

Implicit and Explicit Intent

1) Implicit: l'intent viene inviato e ricevuto da un'app differente

Viene specificata l'attività da fare, **Ha NON L'ATTIVITA' CHE LA DUE FARÀ!**
 Android mostra un chooseer dove l'utente può scegliere con quale attività svolgere l'azione.

2) Explicit: l'intent viene inviato e ricevuto all'interno della stessa APP.

Context-Aware Programming

Le applicazioni che usano il contesto sono chiamate context-aware.

Esse hanno in input, oltre all'input "tradizionale" dell'app, anche il contesto.

Il contesto è importante per le app, in modo da supportare

l'interazione human-computer e viceversa correttamente, in base all'env.



Context

Il contesto è una qualsiasi info che può essere usata per caratterizzare la situazione di un'entità. Un'entità può essere una persona, luogo, oggetto rilevante nell'interazione tra l'utente e l'applicazione, includendo anche l'utente e app stessi.

Esso può essere:

- Fisico: Misurato da sensori HW
- Logico: Acquisito monitorando la user's interaction con l'applicazione o derivato da un contesto fisico.

Uso del contesto

Ci sono tre usi comuni:

- 1) Presentare info e servizi a un utente (es: lista dei monumenti importanti mentre l'utente si muore)
- 2) Esecuzione automatica di un servizio
- 3) Taggare le informazioni per poi restituirlle successivamente (es: Humidity durante una corsa)

Design di applicazioni context-aware

- 1) Specification: Quali comportamenti l'app avrà e in quale situazione ogni comportamento verrà eseguito
- 2) Acquisition: Determinare HW e SW sensori richiesti, usare l'API per comunicare con essi, salvare il contesto e combinarlo con altri contesti
- 3) Delivery: Specificare come il contesto dovrebbe essere dato dai sensori all'applicazione.
- 4) Reception: Analizzare i contesti ricevuti per determinare quali comportamenti context-aware eseguire, e succ. eseguire le azioni

Costruire le app context aware

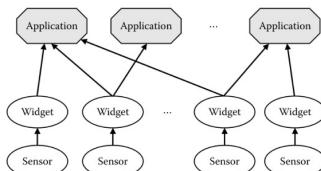
Ci sono tre alternative principali:

- 1) No support
- 2) Widget o obj-based system
- 3) Blackboard-based system

Widget system

Viene usato un context widget, il quale è un SW component che fornisce alle app l'accesso alle context info.

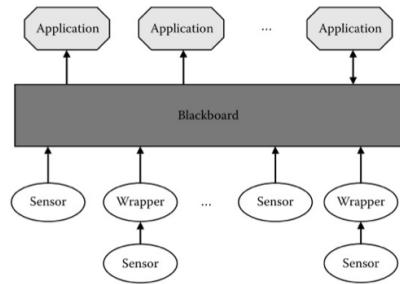
Essi prendono i dati dal sensore con una interfaccia uniforme, in modo da nascondere quale sensore viene usato all'app. (il tipo non impatta il design dell'app.) e per fornire all'app solo le info utili.



Blackboard

In un sistema blackboard-based i componenti possono: piazzare la info all'interno dello storage system, leggerla o rimuoverla. le blackboards possono avere la capacità di notificare i componenti quando l'info di interesse viene aggiunta ad essa.

Problema: la quantità di dati nel tuple space cresce e cercare per una specifica tupla diventa complicato.



Problemi nel costruire applicazioni context-aware

1) Il contesto è solo un proxy per lo human intent

la context info è solo un proxy per un human intent, non è detto però che l'human intent venga capito correttamente.

Solutions:

a) Aggiungere del contesto addizionale per capire lo human intent

b) L'app. dovrebbe chiedere conferma se la sicurezza su quello che ha fatto è al di sotto di una soglia.

2) Context Ambiguity

Context sensors possono catturare i dati incoerentemente, fallire o essere insicuri di ciò che hanno raccolto.

Context influencing systems possono inaccutamente raggiungere conclusioni su una situazione.

Solutions:

1) Combinare più risorse sullo stesso context type

2) Il contesto non cambia randomicamente, ma mostra della coerenza nel tempo.

3) Privacy

I dev delle app context-aware devono assicurare che la privacy di un utente sia mantenuta e che le info non vengano usate impropriamente.

4) Intelligibility of Applications

Il contesto è un input implicito, è difficile per un utente capire perché è stata svolta una certa azione nell'applicazione in uno specifico momento.

Si può fornire agli utenti un feedback per indicare che alcune azioni sono state fatti e il perché.

Benefici di usare il contesto

- 1) Applicazioni pro-attive
- 2) Filtrano le info non rilevanti
- 3) Permettono la creazione di ambienti intelligenti
- 4) Riducono il carico cognitivo sugli utenti

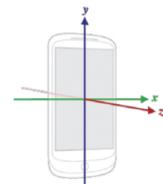
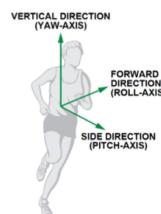
Activity Recognition and Sensors

In questa sezione verrà analizzato come viene svolto il riconoscimento di una attività e quali sensori vengono usati.

Step Analysis

Camminare / correre risulta in un movimento lungo 3 assi:

- 1) Forward (Avanti)
- 2) Vertical (in verticale)
- 3) Side (di lato)

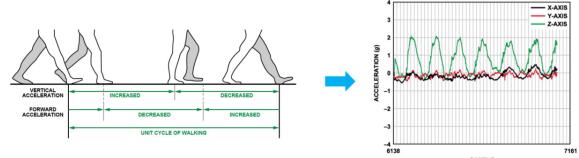


Gli smartphones hanno assi simili e dipendono dall'orientamento del cellulare

La camminata

Durante una camminata, l'accelerazione verticale e in avanti cambia durante le varie fasi della camminata.

Quotidiane, il camminare causa periodicamente delle grosse spike in uno degli assi di accelerazione.



l'asse dipende da come è orientato il telefono.

Step Detection algorithm

Un passo è indicato dal superamento di una determinata soglia e definito come negative slope ($\text{sample}_i < \text{sample}_{i-1}$) quando la smoothed waveform supera una soglia dinamica.

1) Smoothing: Viene ridotto il rumore del segnale usando la media dei valori in una certa finestra

2) Dynamic threshold detection: Dobbiamo adattare la soglia di rilevazione di un passo, cioè quando viene superata, in base all'orientamento del cell e l'andatura.

a) Vengono tracciati i valori minimi e massimi ogni 50 samples.

b) Viene calcolata la soglia dinamica come: $(\max + \min)/2$

Problema: Alcune vibrazioni potrebbero essere contate come passi



Soluzione: Tenere in considerazione la periodicità della camminata/corsa.

1) Assumere che una persona:

a) Correndo fa 4 passi al secondo, cioè impiega 0.25 s per un passo

b) Camminando lentamente fa 1 passo al secondo

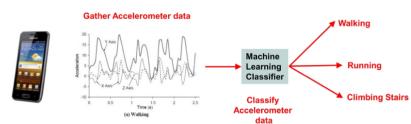
2) Eliminare i passi (negative crossings) che avvengono fuori dal periodo [0.25s - 1s], cioè considerare come passi solo quando la soglia viene superata con un intervallo compreso tra i due valori.

Activity Recognition

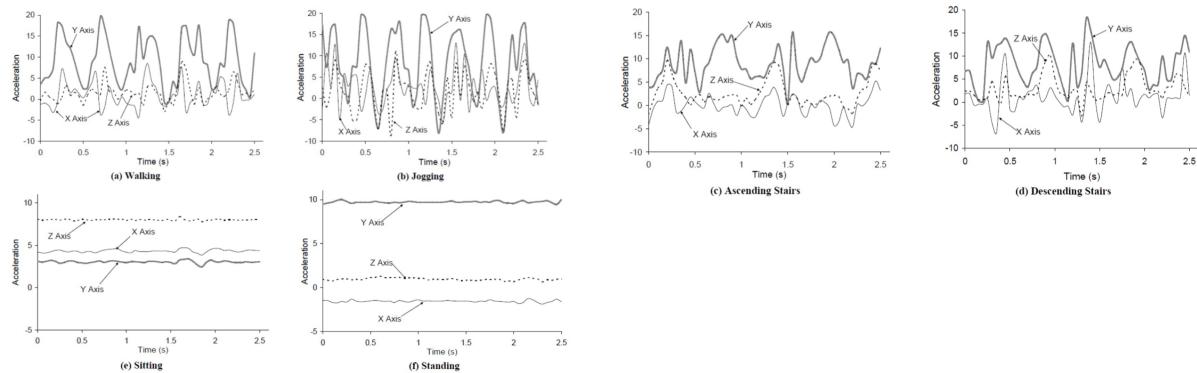
Tipicamente per svolgere un activity recognition task viene usato un ML classifier per classificare i segnali dell'utente catturati dall'accelerometro.

Activity Recognition Applications :

- 1) Fitness tracking 3) Fall detection
- 2) Health Monitoring 4) Context aware behavior ...



Accelerazione delle diverse attività



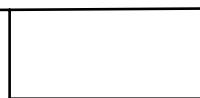
Android Sensors

Android fornisce una API per accedere ai sensori e ottenere i dati con una interfaccia uniforime
In generale le applicazioni:

- 1) Ottengono una lista dei sensori disponibili
- 2) Registrano dei listeners (uno o più)
- 3) Processano i dati tramite callbacks

Essi possono essere caratterizzati in:

- 1) Environmental sensors : temperatura, barometro, luce, umidità
- 2) Motion / Orientation sensors : Usati per rilevare i movimenti



Sensori:

- Accelerometer: returns acceleration
 - Gyroscope: returns angular velocity
 - Gravity: returns direction of gravity, useful for understanding device orientation
 - Linear acceleration: returns acceleration - gravity
 - Rotation vector: returns device orientation
 - Significant motion: used as a trigger to understand device is picked up
 - Step detection: an event for each step
 - Step counter: returns the cumulative number of steps since reboot
 - Magnetic field: amount of geomagnetic field along 3 axes
- } HW
- } SW o HW

- a) Acceleration: include la gravità

- Case 1:* • Standing still on a table, face up: $acc_z = +9.8m/s^2$
- Case 2:* • Free fall: $acc_x=0, acc_y=0, acc_z=0$

- b) Gravity: viene stimata tramite un low pass filter o tecniche più complesse

- c) Linear Acceleration: Non include la gravità \Rightarrow Linear acceleration = acceleration - gravity

- Case:* • Standing still on a table: 0 on all axes

d) Gyroscope : Riconna la velocità angolare

e) Rotation Vector:

- Goal: compute azimuth, pitch, roll
 - Azimuth: angle around z-axis
 - Pitch: angle around x-axis
 - Roll: angle around y-axis
- Based on sensor fusion: gyroscope, accelerometer, geomagnetic field
 - Accelerometer can provide information about pitch and roll because of gravity, but not about azimuth
 - Geomagnetic field can provide information about azimuth because of magnetic north
 - Gyroscope provide info, drift eliminated by means of the other sensors
- Returns orientation with respect to global reference frame
- Orientation: rotation needed to transform a reference system into the other one
- Orientation information can be represented as a vector (x, y, z) which identifies the rotation axis + an angle
- Android's rotation vector returns quaternion representing same information
 $(x \cdot \sin(\theta/2), y \cdot \sin(\theta/2), z \cdot \sin(\theta/2))$
- Commonly used in maps/games

Finebase

È una piattaforma di sviluppo mobile che offre diversi servizi:

Firebase Cloud Messaging

Problema: Gli smartphones sono solitamente dietro le NAT che non permettono di raggiungere gli smartphones se non sono loro ad iniziare la conversazione.

- Firebase Cloud Messaging
- Firebase Realtime Database
- Authentication
- Crash and performance analytics
- Cloud-based content hosting
- Remote configuration
- Advertisement

Soluzione: Firebase Cloud Messaging è una piattaforma di messaggistica che permette di mandare i messaggi agli smartphone.

Esa può notificare una client app che una nuova email/mex è disponibile

Posizione e Happe

Per poter localizzare uno smartphone si può usare:

- 1) Fuori edifici: GPS (più accurato)
- 2) Dentro edifici: WiFi o cell tower signals

GPS

la posizione di una qualsiasi posizione sulla terra è descritta in termini di <latitudine, longitudine>.

Il GPS è ragionevolmente accurato, ma:

- 1) Richiede una "linea di rista" tra il satellite e il receiver
- 2) Funziona solo fuori dagli edifici
- 3) Consuma molta batteria

Wi-Fi location Finger printing

È un sistema basato sul Wi-Fi per rilevare la posizione indoor.

Idea: In ogni posizione (x,y) l'insieme di WiFi Access Points e la loro potenza è unica.

Location	AP1	AP2	AP3
(X, Y)	55	43	65

Quindi lo scopo è stimare la posizione del device basandosi sulla combinazione:

Wi-Fi AP visible + signal strengths

Questa procedura necessita di una tabella che contiene delle tuple pre-registrate.

- 1) Google costruisce e salva questo DB (APs + signal strength) ad ogni (x,y) location.
- 2) I valori osservati non saranno esattamente uguali, quindi viene scelto quello con minor distanza.

Pre-recorded signal strength				
Location (X, Y)	AP1	AP2	AP3	AP4
(10, 33)	32	43	54	65
(42, 21)	76	11	4	32
(5, 12)	-	22	7	43
(65, 3)	16	19	-	5
(8, 21)	2	39	25	17
...

Observed signal strength				
	AP1	AP2	AP3	AP4
	74	11	5	31

=> Oss: Al posto degli AP possono essere utilizzate le cell towers

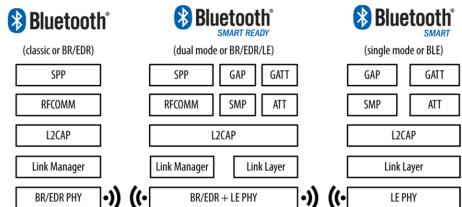
Come costruire la tabella?

I dispositivi con GPS e WiFi access contemporaneamente costruiscono la tabella inviando i dati a uno repository di terze parti o a Google. (Processo conosciuto come "War Driving")

Bluetooth Low Energy (BLE)

BLE è un sottosistema leggero delle specifiche principali del bluetooth 4.0, con l'obiettivo di disegnare un radio standard con il più basso consumo di energia possibile, ottimizzabile per i bassi costi e con una bassa larghezza di banda.

Per poter interagire con il bluetooth classico, il dispositivo più potente necessita di uno stack doppio.



Source: Getting started with Bluetooth Low Energy

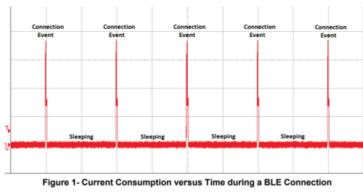
I dispositivi che usano il BLE sono solitamente basati sul SoC (system-on-a-chip).

Throughput

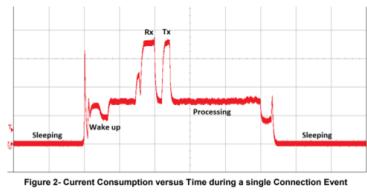
Connection event: Evento nel quale i dati vengono scambiati e i device subito dopo tornano in idle state per salvare energia.

Connection interval: L'intervallo tra due connection event consecutivi;

- 1) throughput più piccolo di 15 kb/s essendo che consumerebbe velocemente la batteria
- 2) Spegnere la radio quando possibile, per più tempo possibile.
- 3) Dati trasmessi in "brevi raffiche" durante i connection events
- 4) Connection events separati il più possibile per salvare energia
- 5) 7.5 ms - 4 ms connection interval:
 - a) Piccolo intervallo = Maggiore throughput
 - b) Grande intervallo = Save energy



Consumo energetico in una connessione BLE



Consumo energetico in un singolo evento di connessione

Transmission Range

La potenza di trasmissione può essere configurata.

Il range massimo è 30m ma solitamente vengono usati 2-5m per salvare connente.

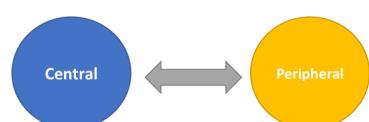
Versioni recenti della BLE (5.2) permettono di adattare dinamicamente il Tx power in base alla distanza del receiver dal transmitter.

Tipi di comunicazione

1) Broadcaster / Observer: Unidirezionale, comunicazione connection less

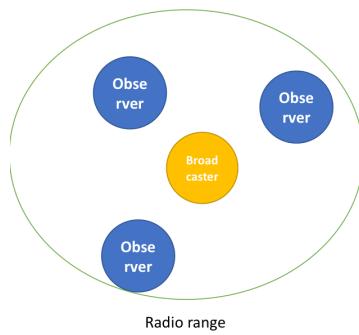


2) Central / Peripheral: Bidirezionale, small devices che si connettono a un dispositivo centrale



Broadcaster / Observer communication

Broadcaster: Invia advertising packets periodicamente a chiunque voglia riceverli (Nel radio range)



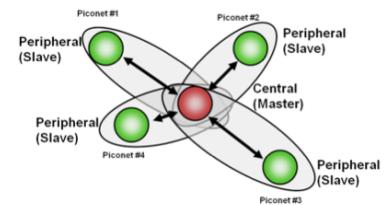
Observer: Scansione ripetutamente le frequenze per ricevere ogni advertising packets che vengono trasmessi.

Central / Peripherals

La connessione è permanente, vengono scambiati periodicamente dei pacchetti tra i due dispositivi.

Central (Master): Scansiona ripetutamente le frequenze per ricevere i connectable advertising packets e avviare la connessione. Successivamente, gestisce il tempo e inizia lo scambio periodico di dati.

Peripheral (Slave): Periodicamente invia connectable advertising packets. Accetta le connessioni in entrata.

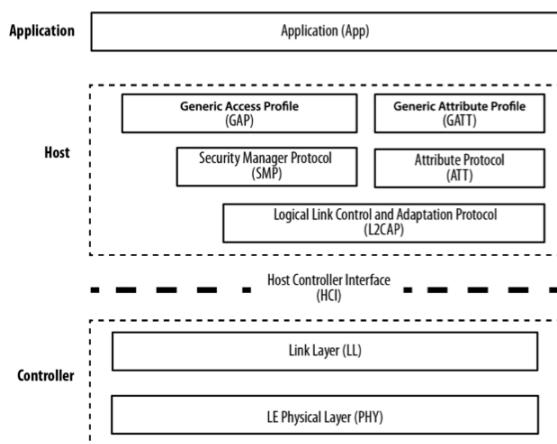


Quando connesso al master, segue il suo timing e scambia i dati con esso

Osservazioni:

- 1) la topologia può essere complessa
 - a) Un dispositivo può agire come central/peripheral allo stesso tempo
 - b) Un central può essere connesso a più peripheral nodes
 - c) Un peripheral può essere connesso a più central nodes
- 2) Può essere più energy efficient del obs/broadcast, dipendentemente dalla comm. behavior
- 3) Può estendere il delay tra le connessioni o inviare i dati solo quando nuovi valori sono disponibili
- 4) I peers sanno quando la prossima connessione avrà luogo, permettendo alla radio di essere spenta più a lungo, risparmiando energia.

BLE Stack



Physical layer

Usa la 2.4 GHz ISM band divisa in 40 canali:

- 37 usati per i connection data
- 3 canali (37,38,39) usati come advertising channel per impostare le connessioni e mandare i broadcast data.

Frequency hopping spread spectrum: la radio salta tra i canali ad ogni connection event usando il seguente schema: $\text{next_channel} = (\text{channel} + \text{hop}) \bmod 37$

Il valore del salto (hop) viene comunicato quando la connessione viene stabilita.

Link layer

Il link layer definisce le seguenti ruoli:

- 1) Advertisers: Il device che invia gli advertising packets
- 2) Scanners: Il device che scansiona per ricevere gli advertising packets
- 3) Master: Device che initializza e gestisce la connessione
- 4) Slave: Device che accetta una richiesta di connessione e segue il timing del master.

Esso inoltre definisce il formato dei frame e che il bluetooth address è di 6 bit, il quale identifica univocamente un dispositivo tra i peers.

Public Device Address: È un indirizzo fisso, generato dall' IEEE

Random Device Address: Indirizzo che può essere programmato sul device o che cambia dinamicamente.

Gli advertisements vengono inviati con un fixed rate definito dal "advertising interval", il quale va in un range tra 20ms ai 10.24s. Essi vengono ricevuti con successo dallo scanner solo quando sono randomicamente sovrapposti (ad inviati e scanner in funzione).

Advertiser/scanner (Broadcast / Observer)

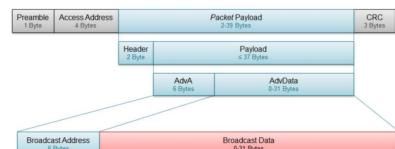
Gli advertising packets standard contengono 31-byte di payload.

Il payload include i dati che descrivono il broadcaster e le sue capacità.

Può anche includere qualsiasi informazione personalizzata che vuoi broadcastare agli altri dispositivi.

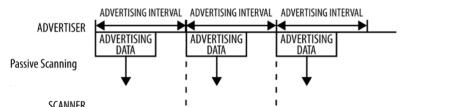
Se 31-byte di payload non sono abbastanza, viene inviato un secondo frame di 31-bytes (62 totali).

Lo scanner può comportarsi in due modi:



1) Passive scanning: lo scanner ascolta per ricevere gli adv packets,

e non avverte l'advertiser che uno o più packets sono stati ricevuti.



2) Active scanning: lo scanner invia una Scan Request dopo aver ricevuto un adv packet.

L'advertiser risponde con uno Scan Response packet.

La comunicazione è comunque unidirezionale.

Connessione

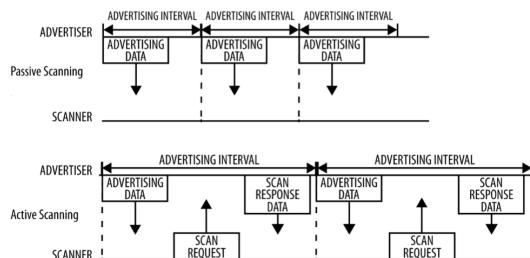
4 data packets sono usati per trasportare dati dell'utente bidirezionalmente tra master e slave.

Sono usabili data payloads da 27 bytes, ma protocolli additionali limitano a 20 bytes per pacchetto.

Il link layer è affidabile:

1) Tutti i pacchetti ricevuti sono checkati a confronto con un 24-bit CRC

2) le richieste sono richieste quando "l'error checking" rileva una connection failure



L2CAP (Logical Link Control and Adaptation Protocol)

Funzionalità principali:

1) Multiplexing / Demultiplexing dai/ai layer superiori

2) Fragmentation / reassembling

Un L2CAP packet header prende fino a 6 bytes, il che significa che l'user payload length effettivo è di $27 - 6 = 21$ bytes.

Interagisce con i layer superiori:

1) Attribute Protocol (ATT)

2) Security Manager Protocol (SMP)

ATT (Attribute Protocol)

È un protocollo client/server stateless basato su attributi presentati da un device.

- Il cliente richiede i dati dal server
- Il server invia i dati al client

I server contengono dei dati organizzati in forma di attributi.

Ogni attributo ha:

- 1) Un "attribute handle" di 16bit, cioè un identificativo usato per accedere al valore dell'attributo.
(row number in una tabella)
- 2) Un Universally unique identifier (UUID): Specifica tipo e natura dei dati contenuti nel valore.
(attribute type)
- 3) Un insieme di permessi
- 4) Un valore : Dipende dall'uuid (attribute type)

	Heart Rate Profile	Handle	Type of attribute (UUID)	Attribute permission	Attribute value
att n 1	Service Declaration	0x000E	Service declaration Standard UUIDservice 0x2800	Read Only, No Authentication, No Authorization	Heart Rate Service 0x180D
att n 2	Characteristic Declaration	0x000F	Characteristic declaration Standard UUIDcharacteristic 0x2803	Read Only, No Authentication, No Authorization	Properties (Notify) Value Handle (0x0010) UUID for Heart Rate Measurement characteristic (0xA37)
att n 3	Characteristic Value Declaration	0x0010	Heart Rate Measurement Characteristic UUID found in the Characteristic declaration value 0x2A37	Higher layer profile or implementation specific.	Beats Per Minute E.g "167"

Security Manager Protocol

Definisce le procedure di sicurezza.

- 1) Pairing
 - a) Una chiave di sicurezza per criptare temporanea viene generata.
 - b) Si trova a un link sicuro e crittato
 - c) La key temporanea non viene salvata e non è riutilizzabile per le connessioni future
- 2) Bonding

- a) pairing + la generazione e scambio di secure keys permanenti
 - b) salvato in memoria non volatile e creating un bond permanente tra due device
 - c) permette la creazione di un secure link nelle connessioni seguenti senza dover performare un bonding again.
- 3) Encryption re-establishment (dopo il bonding)
- a) le chiavi generate in bonding sono salvate su entrambi i lati della connessione
 - b) definisce come usare le chiavi nelle connessioni seguenti per ri-stabilire una connessione sicura e criptata senza dover passare dal pairing e bonding again.

Pairing

Alcune procedure di pairing:

- 1) Just works: viene generata una STK su entrambi i sides, basandosi sui pacchetti scambiati in testo in chiaro
 - Debole ai man-in-the-middle attacks
- 2) Passkey display: uno dei peers displaya una six-digit passkey generata random, all'altro side viene chiesto di inserirla
 - Protezione ai MITM attacks
- 3) Out of Band: dati aggiuntivi vengono trasferiti senza l'uso delle BLE, ad esempio NFC
 - Protezione ai MITM attacks

GAP e GATT

GAP: definisce i ruoli, procedure per inviare i dati, discover devices, stabilire e gestire le connessioni e negoziazione security levels

- 1) Ruoli: Broadcaster / observer, central / peripheral
- 2) Discoverable / non discoverable:

- Discoverable: Scanner può inviare una scan request dopo un pacchetto di adv ricevuto
- Non discoverable: Scanner non può inviare una scan request dopo un pacchetto di adv ricevuto

3) Connectable / Non-Connectable:

- Connectable: Lo scanner può iniziare una connessione
- Non-Connectable: Lo scanner non può iniziare una connessione

GATT: Define un data model e procedure per discover, read, write e push i dati

- I dati sono organizzati in servizi
- I GATT client/server ruoli dipendono esclusivamente dalla direzione nella quale le data request/response vanno
- GATT organizza le info in profiles, services e characteristics

1) **Profile:** Una collezione di servizi che è stata definita o dal Bluetooth SIG o dal peripheral vendor

2) **Service:** Fornisce info attraverso una o più caratteristiche

Ogni servizio può essere riconosciuto da un UUID

- 16 bit : Per servizi ufficialmente adottati dal SIG
- 128 bit : Per i non ufficiali

3) **Characteristics:** Caratteristiche sui dati encapsulati.

Sono identificate da 16 o 128 bit (standard o no)

Esempi:

- HRM:
 - GAP role is peripheral
 - acts as a GATT server when the phone requests data from sensors
 - sometimes act as a GATT client when it requests configuration information from the smartphone
- Smartphone
 - GAP role is central
 - Generally a GATT client, but sometimes can be GATT server

UUID (Universally unique identifier)

Numeri a 128 bit (16 bytes) unici in modo globale.

Essendo che 16 bytes possono essere molti per un BLE packet, è stata aggiunta alle specifiche la forma da 16 bit.

Il formato corto (16 bit) può essere usato solo con UUID definiti nelle BLE specifications

Per ottenere i 128 bit dai 16:

0000XXXX-0000-1000-8000-00805F9B34FB
 ↓
 16 bits

Beacons

Trasmettono piccoli pezzi di informazione e possono essere usati per la localizzazione indoor.

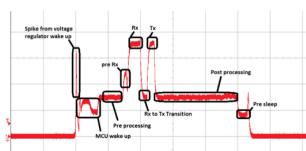
iBeacon: Sviluppato da apple, gli smartphones e tablets fanno delle azioni quando sono vicini agli iBeacons

Eddy Stone: Sviluppato da google, vengono trasmessi gli URL. Notifiche anche senza APP

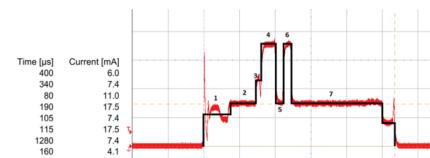
↳ Impiantati dai google beacon.

Power Consumption

- * More in detail
- * MCU wake-up: upon waking up, the current level drops slightly
- * Pre-processing: the BLE protocol stack prepares the radio for sending and receiving data
- * Pre-Rx: the CC2541 radio turns on in preparation Rx and Tx
- * Rx: the radio receiver listens for a packet from the master
- * Rx-to-Tx transition: the receiver stops, and the radio prepares to transmit a packet to the master
- * Tx: the radio transmitter is set to the master
- * Post-processing: the BLE protocol stack processes the received packet and sets up the sleep timer in preparation for the next connection event.
- * Pre-Sleep: the BLE protocol stack prepares to go into sleep mode



	Time [μs]	Current [mA]
State 1 (wake-up)	400	6.0
State 2 (pre-processing)	340	7.4
State 3 (pre-Rx)	80	11.0
State 4 (Rx)	190	17.5
State 5 (Rx-to-Tx)	105	7.4
State 6 (Tx)	115	17.5
State 7 (post-processing)	1280	7.4
State 8 (pre-Sleep)	160	4.1



$$\text{Average current during connection event} = \frac{(\text{State 1 durat.}) * (\text{State 1 current}) + (\text{State 2 durat.}) * (\text{State 2 current}) + \dots}{(\text{Total awake time})}$$

8.24 mA

- * Final step: calculate average current for the entire connection interval, which takes into account the time during which the device is sleeping

$$\text{Average current while connected} = \frac{(\text{Connection Interval} - \text{Total awake time}) * (\text{Average sleep current}) + (\text{Total awake time}) * (\text{Average current during connection event})}{(\text{Connection Interval})}$$

$$= [(1000 \text{ ms} - 2.675 \text{ ms}) * (0.001 \text{ mA}) + (2.675 \text{ ms}) * (8.24 \text{ mA})] / (1000 \text{ ms}) = 0.0230 \text{ mA}$$

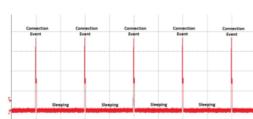
- * The average current consumption while the device is in a connected state is approximately 0.023 mA (23 μA)

- * A coin cell battery CR2032 ha una capacità di 230 mAh

$$\text{Expected battery life} = 230 \text{ mAh} / (0.023 \text{ mA}) = 10000 \text{ h}$$

* Approximately 400 days

- * Between connection events CC2541 goes into low power state PM2 (Power Mode 2)
 - * internal digital voltage regulator is turned off, along with 32 MHz crystal oscillator. The 32 kHz sleep timer remains active while the RAM and registers are retained
- * Current absorbed in sleeping state can be measured: 1 uA



- * Always in connection state, 1 s connection interval, no other sources of consumption

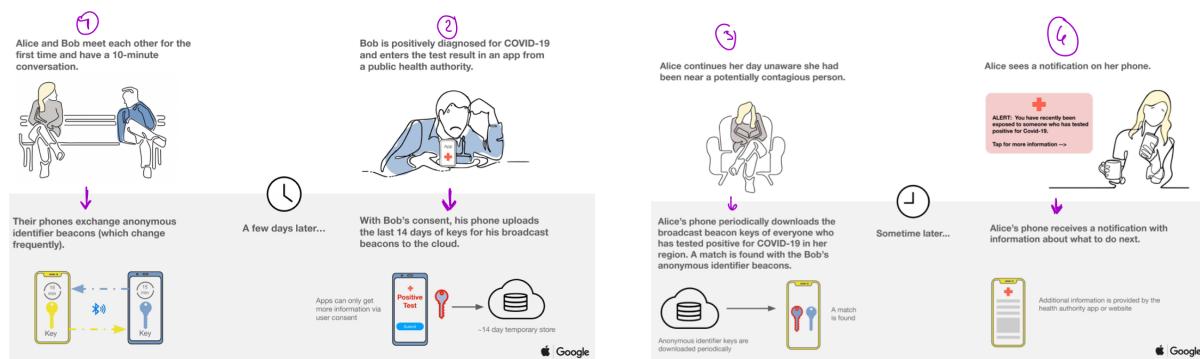
Google & Apple Covid-19 contact tracing (exposure notification)

Le due aziende nel 2020 si sono unite per permettere l'utilizzo del bluetooth per aiutare i governi e le health agencies nel ridurre la velocità di diffusione del virus.

Privacy-safe contact tracing using BLE

- 1) Richiesta esplicita del consenso dell'utente
- 2) Non colleziona info personali sull'identità o la posizione
- 3) La lista delle persone con cui sei stato in contatto non lascia mai il telefono
- 4) Le persone che sono positive non sono identificabili dagli altri utenti, o da Google e Apple
- 5) Sarà utilizzata solo per contact tracing da parte delle health authorities pubbliche

Come funziona?



Exposure Notification Bluetooth Specification v1.2

Protocollo bluetooth che permette l'EN.

L'**exposure notification service** è il veicolo per implementare contact tracing e usa il BLE per la proximity detection degli smartphone vicini.

Esso è un BLE service registrato con il bluetooth SIG con 16-bit UUID 0xFD6F ed è disegnato per permettere il **proximity sensing** dei **Rolling Proximity ID** tra dispositivi.

Il service Data type, con l'UUID di questo servizio, dovrebbe contenere un **128-bit Rolling Proximity ID** e **Associated Encrypted Metadata**, i quali entrambi cambiano periodicamente.

- 1) Temporary Exposure Key : Chiave generata ogni zah per motivi di privacy
- 2) Diagnosis Key : Sottoinsieme di Temporary Exposure Keys che vengono caricate quando il device owner risulta positivo.
- 3) Rolling Proximity Identifier : Un ID derivato dalla temporary Exposure Key e inviato nei bluetooth adv. Esso cambia ogni ~15 minuti per prevenire il wireless tracking del dispositivo
- 4) Associated Encrypted Metadata : Metadati crittati usati per specificare la versione del protocollo e la potenza di trasmissione per una migliore approssimazione della distanza.
- 5) Hashed Message Auth Code : Funzione per la "derivation" delle chiavi

Derivation degli elementi necessari

L'ordine di derivation è il seguente :

Temporary Exposure Key + Discretized time → Rolling Proximity ID Key → Rolling Proximity ID

Il Rolling Proximity ID cambia con la stessa frequenza rispetto al bluetooth randomized Address, per prevenire la "linkability" e "wireless tracking".

Il tempo è discretizzato in intervalli da 10 min che sono enumerati iniziando dall' Unix Epoch time.

EN intervalNumber permette la conversione del tempo corrente in un numero che rappresenta l'intervall nel quale si all'interno.

Other Specifications

Flags			Complete 16-bit Service UUID			Service Data - 16 bit UUID		
Length	Type	Flags	Length	Type	Service UUID	Length	Type	Service Data
0x02	0x01 (Flag)	0x1A	0x03	0x03 (Complete 16-bit Service UUID)	0xFD6F (Exposure Notification Service)	0x17	0x16 (Service Data - 16 bit UUID)	0xFD6F (Exposure Notification Service) 16 bytes Rolling Proximity Identifier Associated Encrypted Metadata 4 bytes

Flag section : BLE discoverable mode dovrebbe essere settato a 1

UUID : 16 bit corrispondente a 0xFD6F

Service Data section : Questa sezione dovrebbe avere due differenti sezioni nel suo payload.

- 16 byte Rolling Proximity ID
- 4 byte Associated Encrypted Metadata che contiene:
 - a) Byte 0 : Versioning
 - b) Byte 1 : transmit power level, usata per migliorare l'appox delle distanze

c) Byte 2 e 3: riservati per usi futuri

Broadcasting

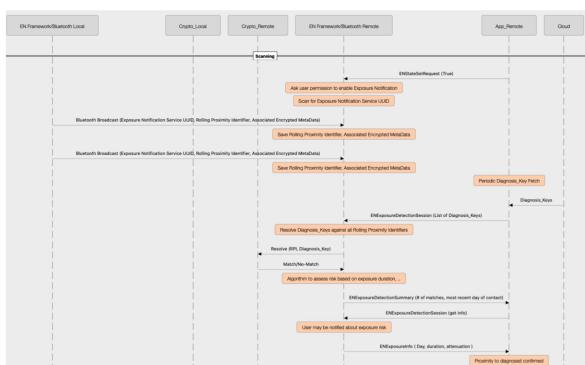
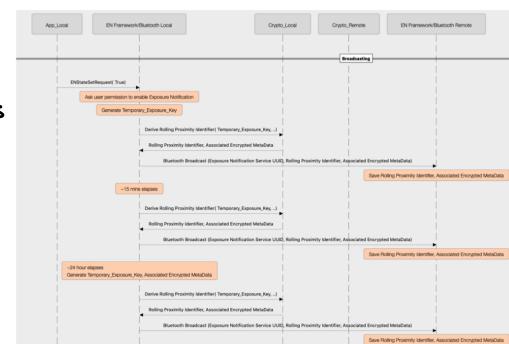
L'advertiser Address type dovrebbe essere random Non-Resolvable.

Sulle piattaforme che supportano il "Bluetooth Random Private Address" con un "randomized rotation timeout interval", l'advertiser "address rotation period" dovrebbe essere un valore randomico, maggiore di 10 min e minore di 20 min.

Broadcast Ynterval: E' raccomandato un intervallo di 200-270 ms

Scanning

I risultati della scan dovrebbero essere timestampati e l'RSSI collaudata per gli adv.



Positive Diagnosis

Quando un utente diventa positivo:

- 1) Un insieme limitato di temporary Exposure Keys e i loro rispettivi ENIntervalNumber, vengono caricati sul "Diagnosis Server".

Questo insieme di temporary Exposure Keys è limitato rispetto a una finestra di tempo nella quale l'utente può essere stato esposto ad altri utenti.

L'insieme limitato delle chiavi è chiamato "Diagnosis Keys".

- 2) Il diagnosis server aggredisce il Diagnosis Keys da tutti gli utenti che sono positivi e distribuisce

questa aggregazione a tutti gli utenti che partecipano alla Exposure Notification.

- 3) Ogni client scanica periodicamente la lista delle nuove "diagnosis key" dal server.

Tramite la temporary exposure key e l'EN intervalNumber associato, ogni client può derivare la sequenza dei Rolling Proximity Identifier che era stato trasmesso bluetooth dagli utenti che erano risultati positivi.

- 4) Successivamente fa un match tra i Rolling Proximity IDs scanzionati e quelli calcolati dalle "Diagnosis Key" per vedere se c'è stato a contatto con un positivo.

Power Consumption And Saving

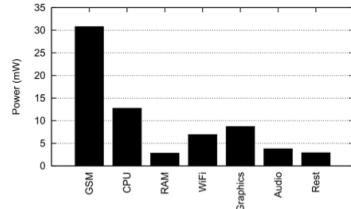
La durata di vita delle batterie è uno dei fattori principali che influiscono in modo negativo sull'esperienza degli smartphone users.

Consumo di uno Smartphone ("ambiente scientifico")

Per calcolare il consumo di ogni componente, sia il supply voltage che la corrente devono essere determinati. La baseline è il dispositivo con nessuna app in esecuzione, dobbiamo considerare due casi:

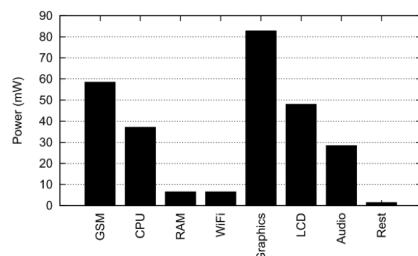
- 1) Suspended: Il processore è in low power state

- GSM module con consumo più alto
- RAM non rilevante anche se a full state
- 69 mW total power



- 2) Idle: Il dispositivo è completamente sveglio e non ci sono app attive.

- Backlight off
- Display-related subsystems hanno il maggior consumo

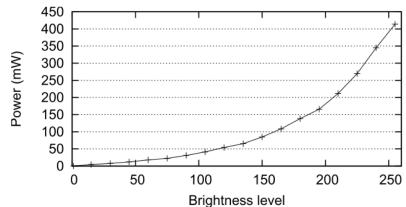


Livello di luminosità

Esso è un intero fra 1 e 255.

Consumo: Min 7.8 mW, max 616 mW

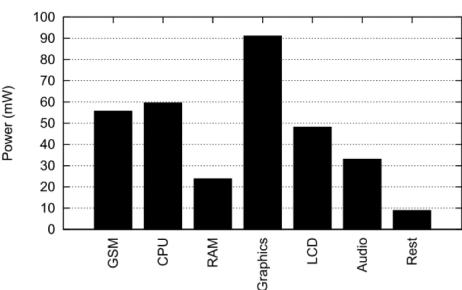
DSS: Sugli schermi LCD, il contenuto mostrato influenza sul consumo.



Audio in playback

Consuma un totale di 320 mW con lo schermo spento.

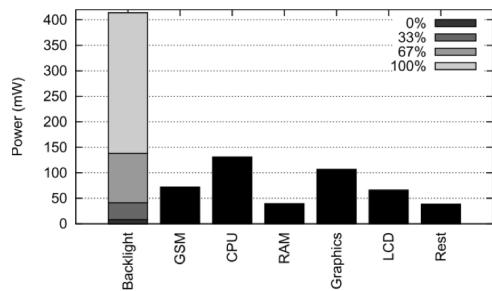
- Audio subsystem = 42% dell'energia consumata
- Mantenere una connessione al GSM richiede tanta potenza
- Muovere i file dall'sd richiede una potenza trascurabile



Video playback

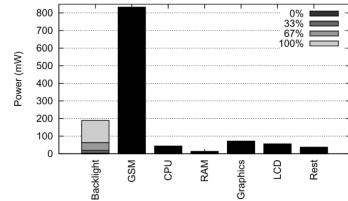
Consuma un totale di 453 mW, senza backlight.

- La CPU è il consumo maggiore oltre allo backlight.
- Il display subsystem è comunque da considerare.
- Muovere i file dall'sd richiede una potenza trascurabile



Phone call

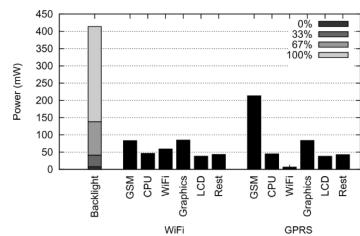
Consumo totale 4056 mW escludendo il backlight.



Browsing web

Consumo totale: 352 mW con il wifi e 429 mW con il GPRS

GPRS consuma più energia rispetto al WiFi con un fattore di 2.5



Analysis of power consumption

Come abbiamo visto, l'energia viene consumata principalmente in:

- 1] Luminosità → Soluzioni: Riduzione luminosità, adattamento automatico, color scheme diversa

2) Connessione con la rete (GSM)

3) Consumo base dei componenti

Consumo di uno Smartphone ("wild ambient")

In questo caso il consumo è misurato in accordo con un metodo meno accurato ma in uno scenario più realistico.

Power Model

Hardware component power draw	Model trigger
CPU	frequency + utilization
GPU	frequency + utilization
Screen	brightness level
WiFi	FSM + signal strength
3G/LTE	FSM + signal strength
WiFi beacon	WiFi status
Cellular Paging	cellular status
SOC Suspension	constant

+) CPU

$$P_{CPU} = P_{B,N_c} + \sum_i^{N_c} u_i \cdot P_{\Delta}(f_i)$$

Dove:

• P_{B,N_c} = Baseline CPU power con N_c processori attivi

• $P_{\Delta}(f_i)$ = Power increment del core i alla frequenza f_i e u_i il suo utilizzo

Misurato in lab e a riunione l'energia di ogni app è stimata usando la logged cpu frequency e utilization

2) GPU : Approssimazione simile alla CPU

3) Screen: Consumo a diversi livelli di luminosità, misurato in lab (contenuto sul display non considerato)

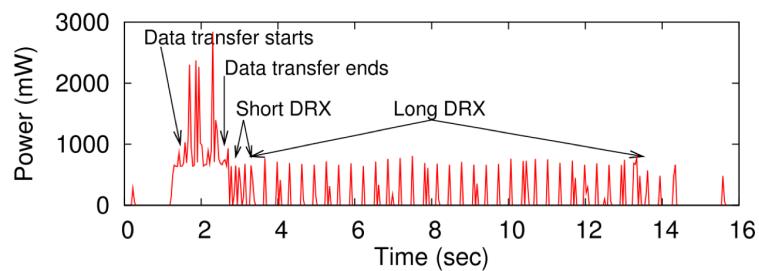
4) 3G / LTE : 4g power model in questo caso è composto da 6 fasi

a) IDLE: l'interfaccia è IDLE quando l'UE non invia/n riceve dati. Periodicamente si sveglia per verificare se ci sono dati buffered nella rete, in entità.

b) CR: Quando l'UE invia o riceve dati, l'interfaccia entra nel Continuous Reception (CR)

state e consuma tanta energia.

- c) Short DRX: Dopo che l'UE finisce il trasferimento dati e diventa IDLE per 200ms, l'interfaccia entra nel short DRX state, dove c'è poco consumo di energia ma si sveglia freq. per verificare se c'è del traffico in entrata.
- d) Long DRX: L'interfaccia entra in long DRX state dopo essere stata per 600ms senza ricevere nessun dato, i wakeup interval diventano più lunghi.



5) WiFi: simile al 3G/LTE

6) WiFi Beacon: WiFi radio needs to search all possible channels until it finds one. For example, the 2.4 GHz band (802.11/bg) has 11 channels and the 5GHz (802.11ac) band has 22 channels. WiFi scanning on these two phones has a duration of 3.4 seconds and average power draw of 64 mA

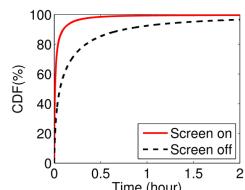
7) Cellular Paging: On a cellular network, the base station periodically broadcasts a message during the 3G/LTE Idle state to signal incoming downlink data or voice call or SMS. A power spike on the phone modem every 1.28s. Average current: 8.3 mA on S3 and 2.3 mA on S4.

8) SoC Suspension: When the CPU and other hardware components are offline, the entire SoC is suspended and draws a constant current: 3.8 mA and 5.1 mA for Galaxy S3 and S4, respectively

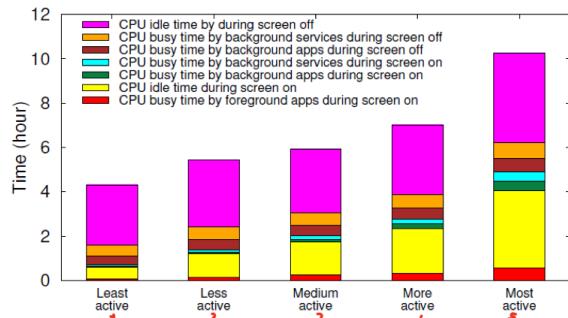
Analisi e Implicazioni

L'utilizzo è stato tracciato tramite l'utilizzo di un'app.

È stato confrontato l'utilizzo su 4520 utenti e 32 giorni su periodi di schermo acceso/spento



Gli utenti sono stati divisi in 5 gruppi in base al tempo totale di schermo acceso (screen-on)



- In foreground, except for games that can be keeping CPU busy while the user is idle, most apps are not using the CPU: users are idle for a significant portion of the time, e.g., reading web pages, emails, or thinking.
- A huge portion of screen-off CPU time is spent idle (50.4%) which should ideally be close to zero.

- Energy breakdown by activities
 - 45.0% of the total energy drain in a day occurs during screen-off periods
 - 21.2% of the total energy drain due to SOC base power, WiFi beacon, WiFi scanning and cellular paging activities during screen-off periods does not contribute to any useful work
 - Out of the 55.0% energy incurred during screen-on periods, 24.0% is spent in screen

- Energy breakdown by components
- Cellular paging vs. WiFi beacon:** The energy drain from cellular paging and WiFi beacon are 12.5% and 2.3%, respectively
- Cellular vs. WiFi:** The energy drain over cellular (LTE and 3G) and over WiFi are 11.7% and 1.3%, respectively

Cosa abbiamo imparato dai due studi?

- Lo schermo consuma una grossa fetta di energia
- WiFi consuma meno del 3%
- CPU è tolle nella maggior parte del tempo
- Un grosso ammontare di energia è speso durante le attività in background

CPUs più efficienti

Il calcolo computazionale è generalmente leggero, ma con alcune spille.

Alcune volte è necessaria più potenza computazionale.

- High Performance CPU: Utente felice ma molto consumo di energia per le cose "comuni"
- Low Performance CPU: Ridotto consumo ma utente poco felice

Multi processing

Possiamo usare più processori per migliorare le performance:

- +) Symmetric Multiprocessor (SMP) : Aumenta le performance usando il parallelismo.

Tutti i processori sono identici.



- 2) Heterogeneous Multiprocessor (HMP) : Aumenta le performance usando il parallelismo.

I processori sono diversi tra loro.



ARM's big.LITTLE (HMP)

È un sistema HMP. Il sistema usa il giusto processore in base alla potenza richiesta.

Usa sia il Dynamic Voltage, sia il Frequency Scaling.

Dynamic Voltage e energy Scaling

Sono tecniche per risparmiare energia che usano:

- +) La relazione lineare tra consumo di energia e frequenza
- 2) La relazione quadratica tra consumo di energia e voltaggio

La relazione è data da:

$$P = C \times V^2 \times f \quad \text{dove:}$$

P: power
C: switching capacitance of the logic circuit
V: operational voltage
f: operational frequency

A bassa frequenza il processore può lavorare a basso voltaggio.

Operating Point = Frequenza e voltaggio usati

Task Migration

Possono essere usati tre tipi di task migration:

- +) Clustered switching
- 2) CPU Migration (in-kernel scheduling)
- 3) Heterogeneous Multiprocessing (global task scheduling)

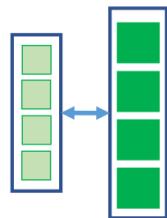
Clustered Switching

L'insieme di processori piccoli e grandi hanno la stessa dimensione.

Solo un singolo cluster può essere attivo in un certo momento.

Quando il carico aumenta, i task vengono trasferiti al big cluster e viceversa.

Il cluster inattivo viene spento.



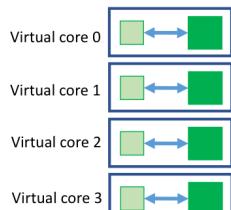
- Not all cores are used
- Simple
- Sub-optimal both in terms of energy and computing power

CPU Migration

L'insieme di processori piccoli e grandi hanno la stessa dimensione.

Quando il carico aumenta, i task vengono trasferiti al big cluster e viceversa.

Il processore RFAFE inattivo viene spento.



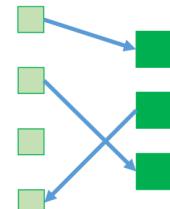
- Not all cores are used
- Simple
- Sub-optimal both in terms of energy and computing power

Global Task Scheduling

L'insieme di processori piccoli e grandi possono avere dimensioni diverse.

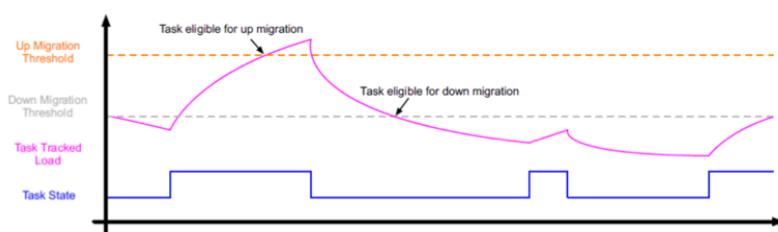
Quando il carico aumenta, i task vengono trasferiti al big cluster e viceversa.

Gli processori inattivi vengono spenti.



- All cores may be in use at the same time
- More complex scheduling decisions

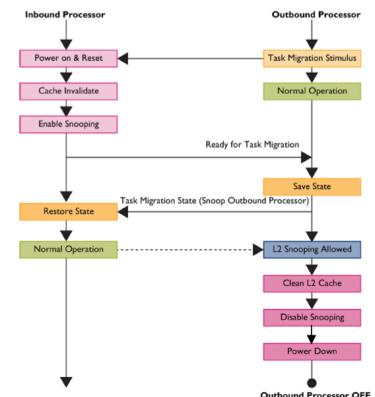
In GTS lo scheduler deve tracciare l'attività level di ogni task



TASK MIGRATION TIME

Per migliorare il tempo delle task migration:

- 1) lo state che è salvato sul outbound processor viene "sprato" e ricreato sull'inbound processor, invece che passare dalla mem. principale.
- 2) la cache di livello 2 dell'outbound processor rimane accesa dopo la task migration per migliorare il "riscaldamento" della cache dell'inbound processor attraverso lo snooping (spia) dei dati



Energy Aware Scheduling (EAS)

Le Completely Fair scheduler mira ad ottimizzare il throughput e performance delle applicazioni interattive.

La power è gestita da due componenti esterni:

- 1) CPU-idle: la CPU può entrare in diversi Low Power Modules chiamati C states.
Più è bassa la potenza, più tempo ci mettono le operazioni.
CPUidle è chiamata quando la CPU è IDLE per selezionare in quale C state entrare.
- 2) CPU-freq: Determina la frequenza dell'op. della CPU
Se l'uso della CPU è sopra una certa soglia la frequenza è aumentata, se è sotto invece viene diminuita.

In EAS lo scheduler è in una posizione migliore per prendere decisioni sulla potenza rispetto alla CPUidle e CPUfreq.

EAS è un energy-aware scheduler usato nei versioni recenti di Android.

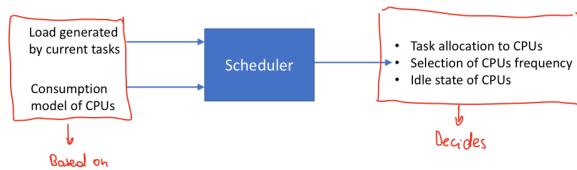
Le principali operazioni sono:

- 1) Un modello delle CPU e cluster fornisce informazioni sui power requirements nei differenti

operational points

- 2) 4 task sono tracciati per valutare il loro impatto sulle CPU
- 3) Quando un task deve essere assegnato alle CPU, la decisione prende in considerazione l'energy impact

EAS opera solo quando c'è capacità inutilizzata, se il carico è alto e non c'è nulla da salvare, il comportamento ritorna allo standard Completely Fair scheduler (CFS)



EAS consumption Model

Il consumo model consiste in:

- 1) Topologia
- 2) Power consumption per ogni operating point
- 3) Power Consumption per ogni c-state
- 4) Wake-up energy cost per ogni c-state
- 5) Il modello contiene i dati per la CPUs e Clusters

EAS Window Assisted Load tracking (WALT)

EAS usa la WALT che è una variazione dell' Entity load tracking (ELT) del CFS.

- 1) In accordo con alcune benchmarks, fornisce alcuni benefit
- 2) Il tempo è diviso in windows
- 3) Il carico generato da un task è valutato per ogni finestra
- 4) Il carico previsto di un task è uguale a
 $\max(\text{load_during_last_window}, \text{avg}(\text{last_N_windows}))$
- 5) La finestra non esiste quando il task è bloccato: Un task pesante uscito dalla sleep è immediatamente classificato come esigente.

Lo scheduler e' ora a conoscenza dell' idle state delle CPU's.

Quando sveglia una CPU, sceglie la CPU nel idle-state più superficiale, minimizzando il wake-up time e energia.

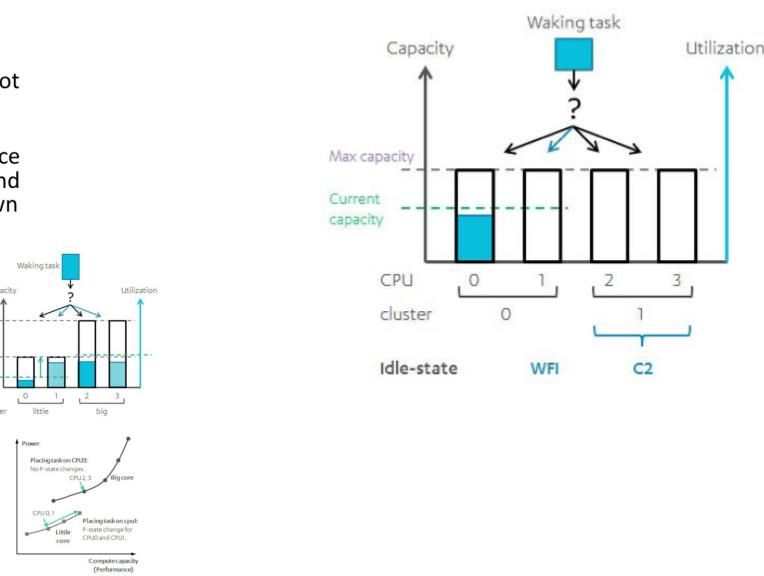
Example:

- new task needs to wake up, but it will not fit on CPU#0 because the current operating point is almost fully utilized
- The new task gets placed on CPU #1 since it is in the shallowest idle state (WFI), and the other cluster remains in C2 shutdown

EAS

- Selection of CPU
- A new waking task can be placed on either of two CPUs - #1 and #3
- EAS estimates the energy costs associated to the two options:
 - CPU#1: operating point must be moved up for both CPU#0 and CPU#1
 - CPU#3: no operating point change, but higher power used

EAS will probably choose CPU#1 because the small additional energy cost of increasing the OPP of CPU#0 (and CPU#1 by implication - since both CPUs are in the same frequency domain in this example) is not significant compared with the better power efficiency of running the task on CPU#1 instead of CPU#3.



Power-Saving ad alto livello

Android include alcuni power-saving mechanisms:

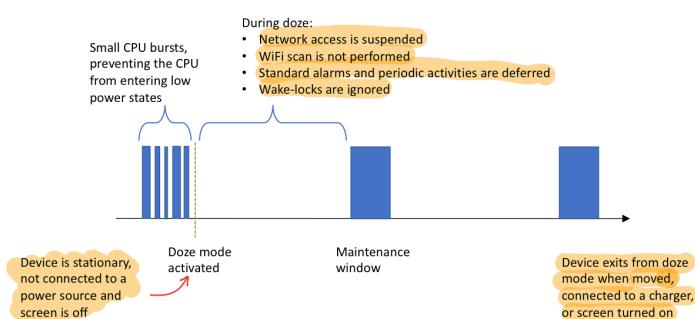
1) Doze Mode

2) App Standby

Doze

Attivati quando il device non è connesso a una fonte di energia e lasciato stationary per un lungo periodo di tempo.

Background CPU e le attività network sono diffuse.



App Standby

Android limita le risorse disponibili alle app dipendentemente su quanta frequente e quanto recentemente sono state usate.

Le App vengono assegnate in dei priority buckets:

- 1) Attivi: App che sono attualmente usate o molto recentemente
- 2) Working: Spesso usate, ma non giornalmente
- 3) Rare: Non frequentemente usate

Quando il device non è connesso alle power supply, le restrictions sono imposte alle app dipendentemente dal loro "bucket".

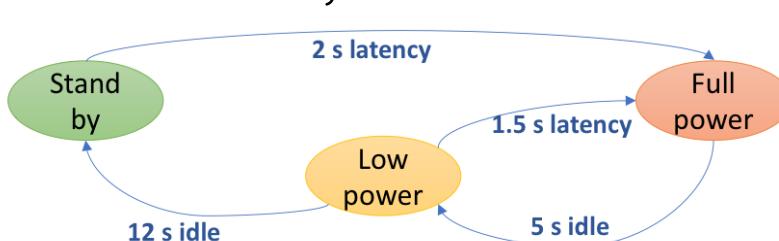
Bucket	Jobs	Alarms	Network	FCM
Active:	No restriction	No restriction	No restriction	No restriction
Working set:	Deferred up to 2 hours	Deferred up to 6 minutes	No restriction	No restriction
Frequent:	Deferred up to 8 hours	Deferred up to 30 minutes	No restriction	High priority: 10/day
Rare:	Deferred up to 24 hours	Deferred up to 2 hours	Deferred up to 24 hours	High priority: 5/day

- Works/Jobs: operations to be executed when specific conditions are met (e.g. network is available, device is charging, timeout)
- Alarms: mechanisms useful to perform time-based operations even when an app is not running.
- Firebase Cloud Messaging: a mechanism useful to send messages to android smartphones (e.g. new data on a server is available)

Reducing energy consumption due to communications

L' energia richiesta dal 3G radio dipende dal suo stato:

- 1) Full power: Una connessione è attiva e i dati possono essere ricevuti/inviai
- 2) Low power: I dati non possono essere trasferiti (consumo $\approx 50\%$), poco tempo per tornare in full power.
- 3) Standby: I dati non possono essere trasferiti, tanto tempo per tornare in full power
(consumo minimo)



Come ridurre il consumo

- 1) Trasferire più dati insieme, ma meno frequentemente



→ Non aggregando



→ Aggregando

È facile quando i dati dipendono dall'app logic, difficile quando dipendono dall'azione dell'utente.

- 2) Cache Data perché usare la rete costa di più che leggere dallo storage locale.