android battery consumption

the what & how

kumar rangarajan

little eye labs

introducing

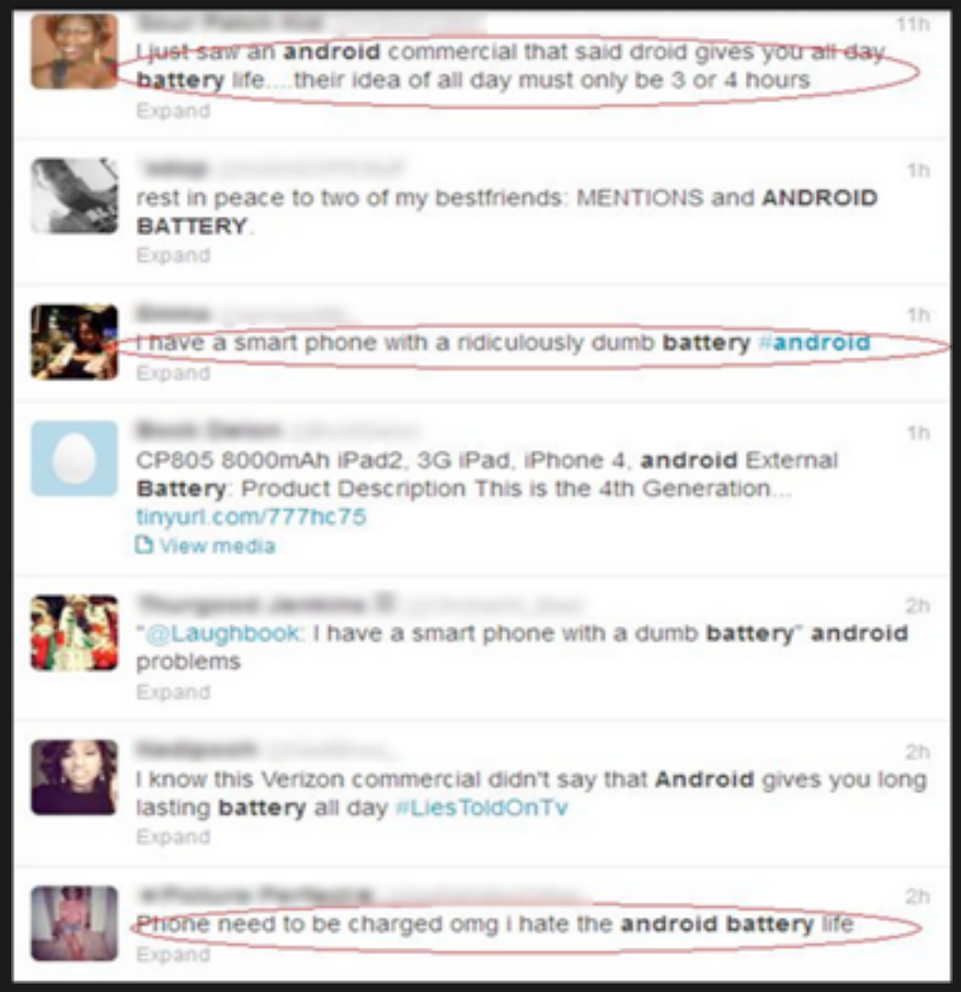
littleEye appInsight

CT scanner for your Android App

why should you care about battery consumption?

- #1 pain point of smart phones users.
- more visible on Android than in iOS
- unlike memory, you can never reclaim excess usage!
- no moore's law for battery capacity
- improving battery capacity is an ongoing research activity, but no current solution
- even then keeping up with consumption would be difficult as apps get more powerful and resource hungry

we code... consumers suffer



Bad battery life is a big problem for

- Users
- Developers
- Ad networks
- Smartphone manufacturers

why should you care about battery consumption?

- #1 pain point of smart phones users.
- more visible on Android world than in iOS
- unlike memory, you can never reclaim excess usage!
- no moore's law for battery capacity
- improving battery capacity is an ongoing research activity, but no current solution
- even then keeping up with consumption would be difficult as apps get more powerful and resource hungry

why should you care about battery consumption?

- #1 pain point of smart phones users.
- more visible on Android world than in iOS
- unlike memory, you can never reclaim excess usage!
- no moore's law for battery capacity
- improving battery capacity is an ongoing research activity, but no current solution
- even then keeping up with consumption would be

why should you care about battery consumption?

- #1 pain point of smart phones users.
- more visible on Android world than in iOS
- unlike memory, you can never reclaim excess usage!
- no moore's law for battery capacity
- improving battery capacity is an ongoing research activity, but no current solution
- even then keeping up with consumption would be difficult as apps get more powerful and resource hungry

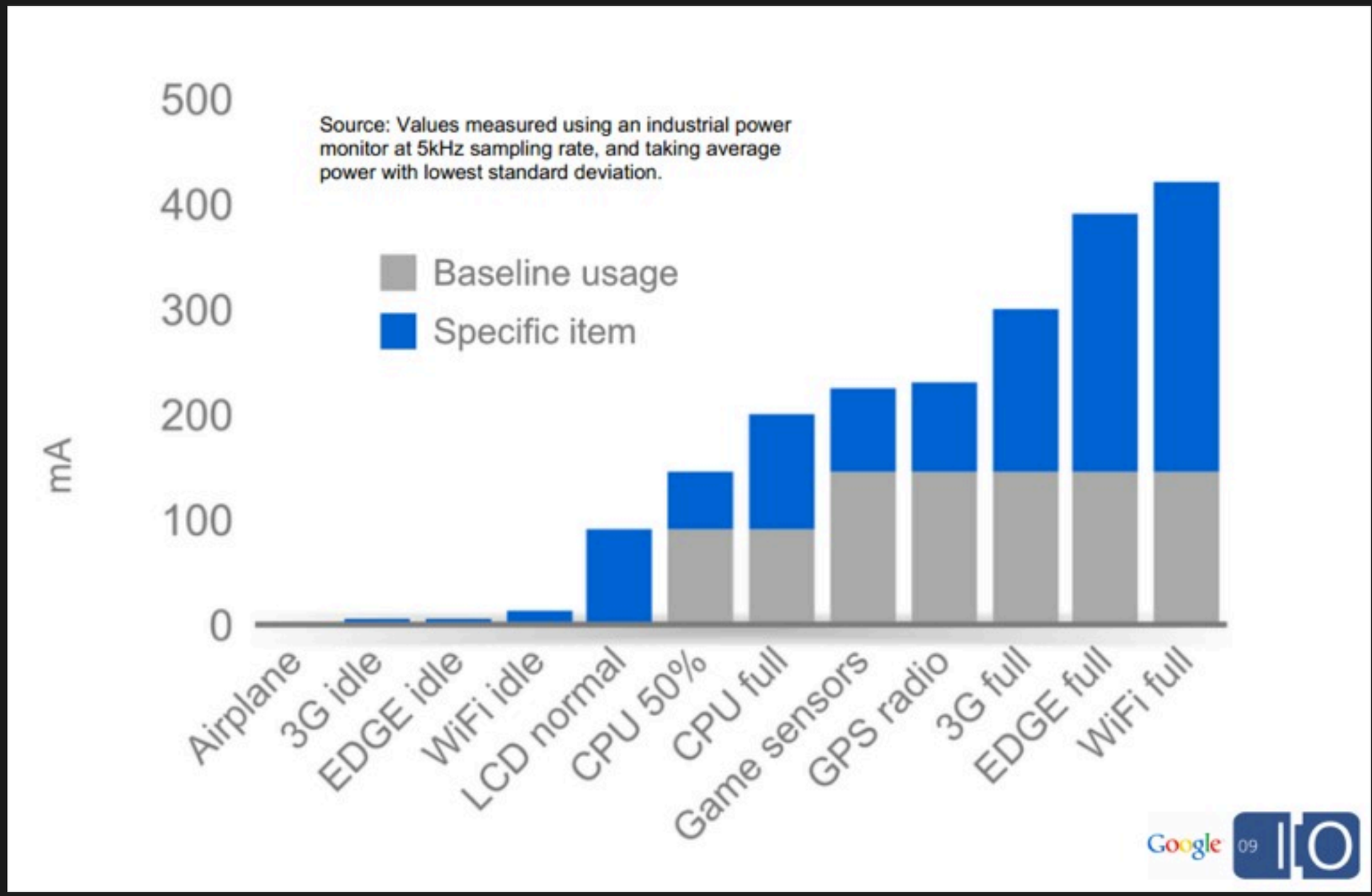
why should you care about battery consumption?

- #1 pain point of smart phones users.
- more visible on Android world than in iOS
- unlike memory, you can never reclaim excess usage!
- no moore's law for battery capacity
- improving battery capacity is an ongoing research activity, but no current solution
- even then keeping up with consumption would be difficult as apps get more powerful and resource hungry

why should you care about battery consumption?

- #1 pain point of smart phones users.
- more visible on Android world than in iOS
- unlike memory, you can never reclaim excess usage!
- no moore's law for battery capacity
- improving battery capacity is an ongoing research activity, but no current solution
- even then keeping up with consumption would be difficult as apps will get more powerful and resource hungry

what consumes power?



originally published in Google IO by Jeff Sharkey

network activity

- EDGE consumes less power for a byte of data transfer, but is slow
- Wifi consumes more power on average, but can do job faster
- 3G consumes more power than Wifi, and typically slower than Wifi.
- 4G consumes more power than 3G, and depending on availability, faster than Wifi

4G > 3G > Wifi > 2G

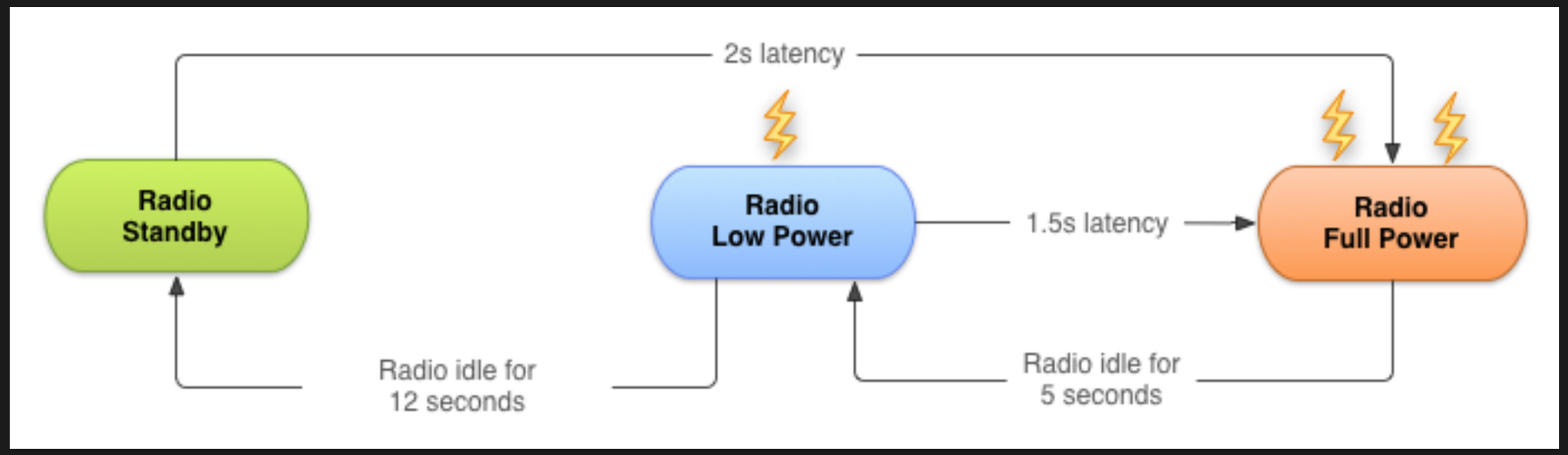
what causes power consumption?

- four primary states -
  - establishing connection or association
  - maintaining association (idle)
  - transmission (two or more sub states)
  - tail state (pseudo state)

	Establish	Maintain	Transmit	Tail
Wifi	High	Low	Low	Low
2G	Low	Low	Low	Low
3G	Low	Low	High	High

derived from http://people.cs.umass.edu/~arun/papers/TailEnder.pdf

more about states

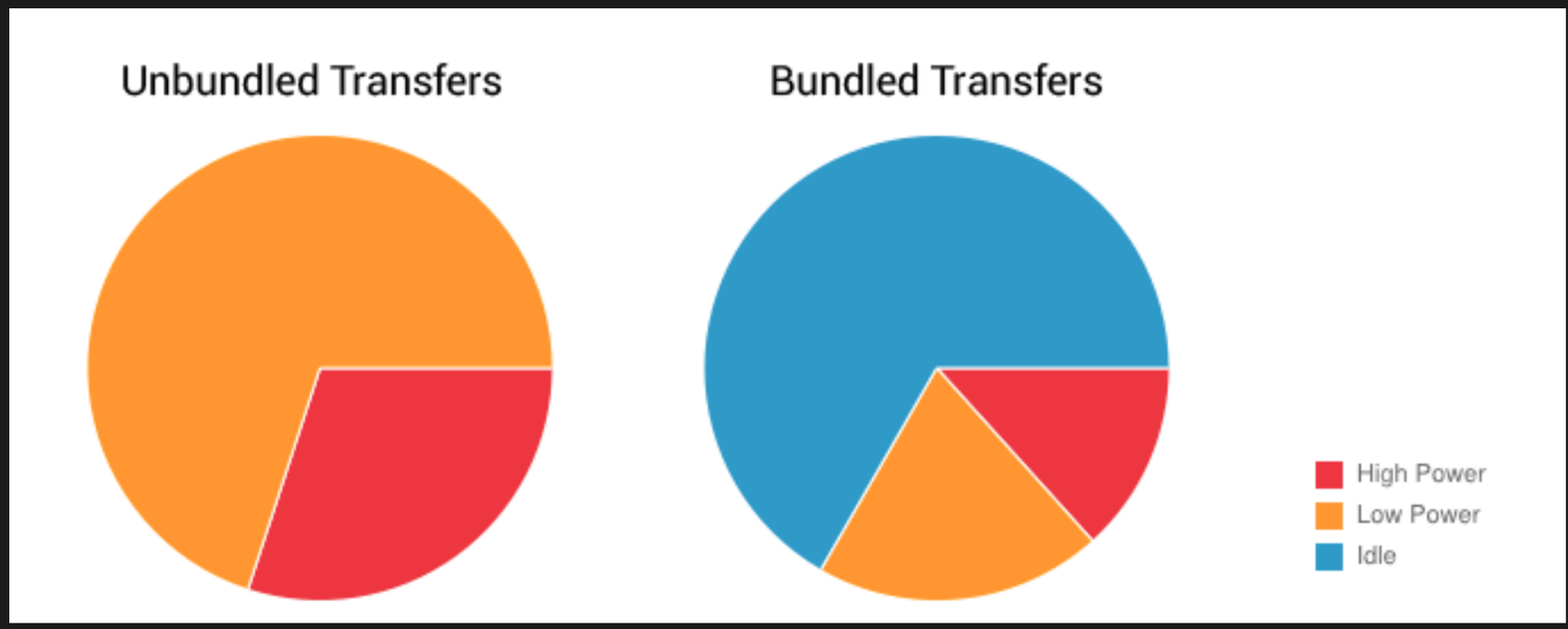


time between state reduction = tail state

how apps can impact state

	number of data operations	duration between operations	size of each operation	time in high/ low state
app1	3	18	1 mb/s	18 + 42 seconds
app2	3	0 (bundled)	1 mb/s	8 + 12 seconds

relative power consumption



Total Current Consumed for 1 minute

app1	78mA
app2	28mA

app1 consumes 3x app2

best practices for network

- prefetch data
  - to avoid switching states frequently, try and read as much data as possible
  - trade-off between too much pre-fetch and battery drain
  - google recommends prefetching data that you will initiate in the next 2 to 5 minutes, and in the order of 1-5 MB size
- batch data
  - do data send/receive in batches rather than on-demand
  - eg: batch analytics information, rather than sending them as they are collected

best practices for network

- prefetch data
  - to avoid switching states frequently, try and read as much data as possible
  - trade-off between too much pre-fetch and battery drain
  - google recommends prefetching data that you will initiate in the next 2 to 5 minutes, and in the order of 1-5 MB size
- batch data
  - do data send/receive in batches rather than on-demand
  - eg: batch analytics information, rather than sending them as they are collected

best practices for network

- detect network connection
  - avoid connection attempts if no network is active
- avoid polling and use GCM when possible
  - avoids multiple connections
  - reduces the number of device state changes
- using inexact timers
  - *AlarmManager.setInexactRepeating()*
  - use *ELAPSED_REALTIME* instead of *ELAPED_REALTIME_WAKEUP*

best practices for network

- detect network connection
  - avoid connection attempts if no network is active
- avoid polling and use GCM when possible
  - avoids multiple connections
  - reduces the number of device state changes
- using inexact timers
  - *AlarmManager.setInexactRepeating()*
  - use *ELAPSED_REALTIME* instead of *ELAPED_REALTIME_WAKEUP*

best practices for network

- detect network connection
  - avoid connection attempts if no network is active
- avoid polling and use GCM when possible
  - avoids multiple connections
  - reduces the number of device state changes
- using inexact timers
  - *AlarmManager.setInexactRepeating()*
  - use *ELAPSED_REALTIME* instead of *ELAPED_REALTIME_WAKEUP*

best practices for network

- use caching
  - avoid redundant downloads
- varying download pattern
  - modify pattern based on connection type
  - download more data per session on faster networks
    - change the aggression of pre-fetch

best practices for network

- use caching
  - avoid redundant downloads
- varying download pattern
  - modify pattern based on connection type
  - download more data per session on faster networks
    - change the aggression of pre-fetch

other tips (from google)

- monitor charge level and state
- monitor and determine the docking state and type
- monitor the connectivity state
- manipulate broadcast receivers on-demand

other tips (from google)

- monitor charge level and state
- monitor and determine the docking state and type
- monitor the connectivity state
- manipulate broadcast receivers on-demand

other tips (from google)

- monitor charge level and state
- monitor and determine the docking state and type
- monitor the connectivity state
- manipulate broadcast receivers on-demand

other tips (from google)

- monitor charge level and state
- monitor and determine the docking state and type
- monitor the connectivity state
- manipulate broadcast receivers on-demand

if your audience has not gone
to sleep yet
&
organizers have...

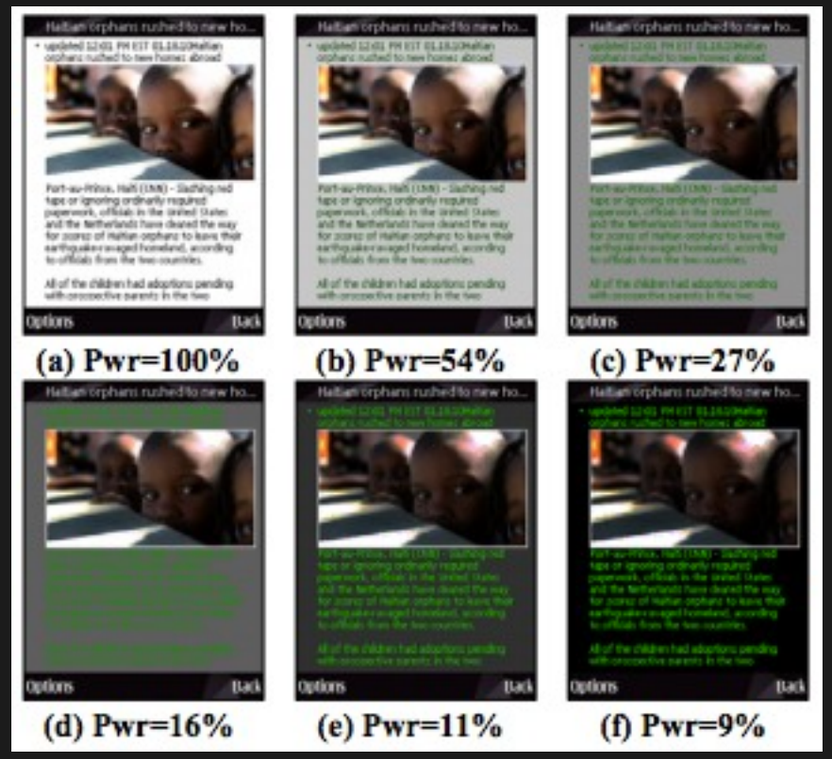
screen

- color matters!
  - esp on OLED screens
- darker the color, lesser the consumption
- brightness levels have more impact
  - refer to - http://stackoverflow.com/questions/5032588/cant-apply-system-screen-brightness-programmatically-in-android
  - programmatically reduce brightness if its suits your app/activity

Chameleon: Color Transformation on OLED Displays

screen

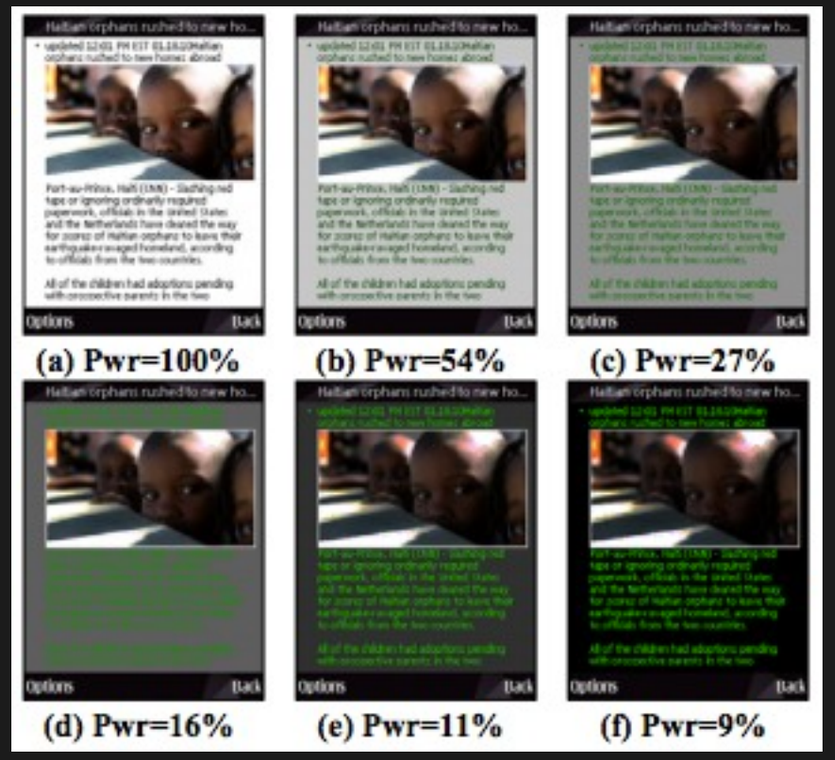
- color matters!
  - esp on OLED screens
- darker the color, lesser the consumption
- brightness levels have more impact
  - refer to - http://stackoverflow.com/questions/5032588/cant-apply-system-screen-brightness-programmatically-in-android
  - programmatically reduce brightness if its suits your app/activity



Chameleon: Color Transformation on OLED Displays

screen

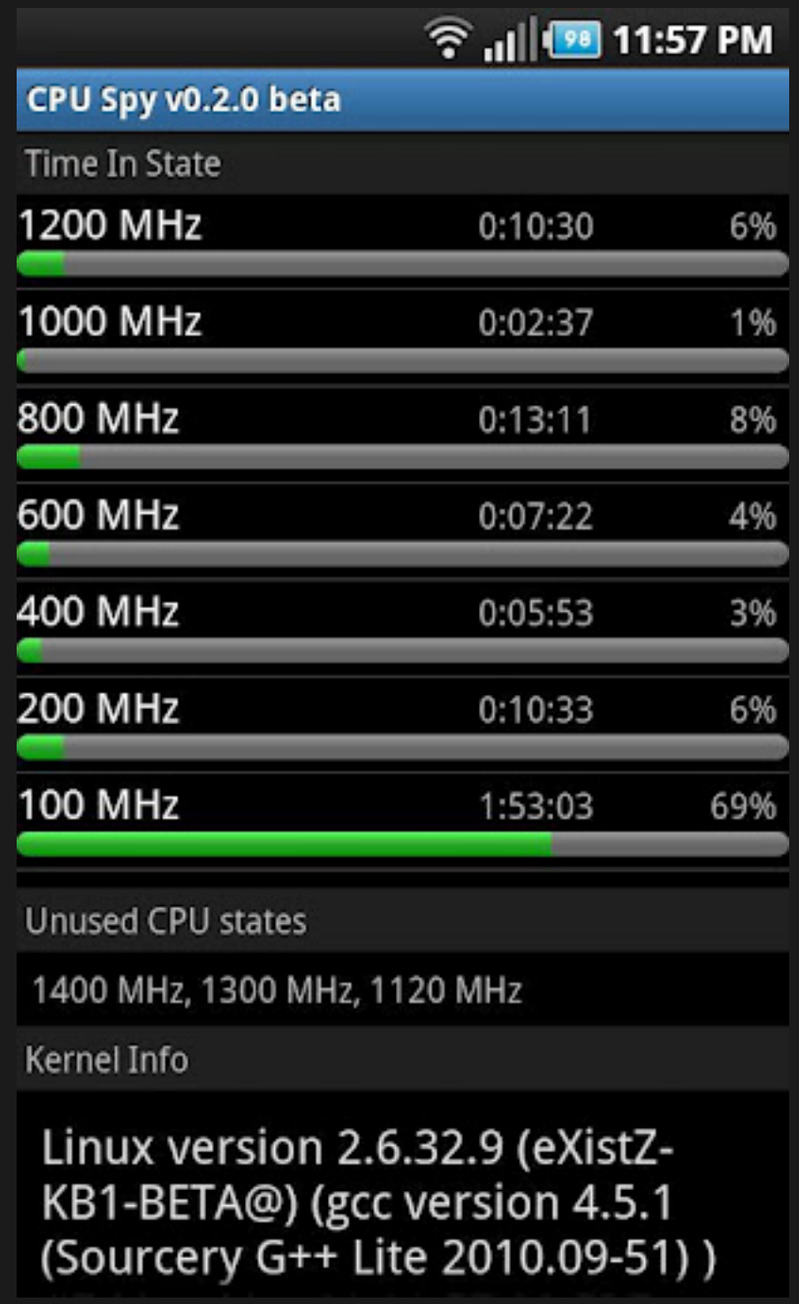
- color matters!
  - esp on OLED screens
- darker the color, lesser the consumption
- brightness levels have more impact
  - refer to - http://stackoverflow.com/questions/5032588/cant-apply-system-screen-brightness-programmatically-in-android
  - programmatically reduce brightness if its suits your app/activity



Chameleon: Color Transformation on OLED Displays

cpu

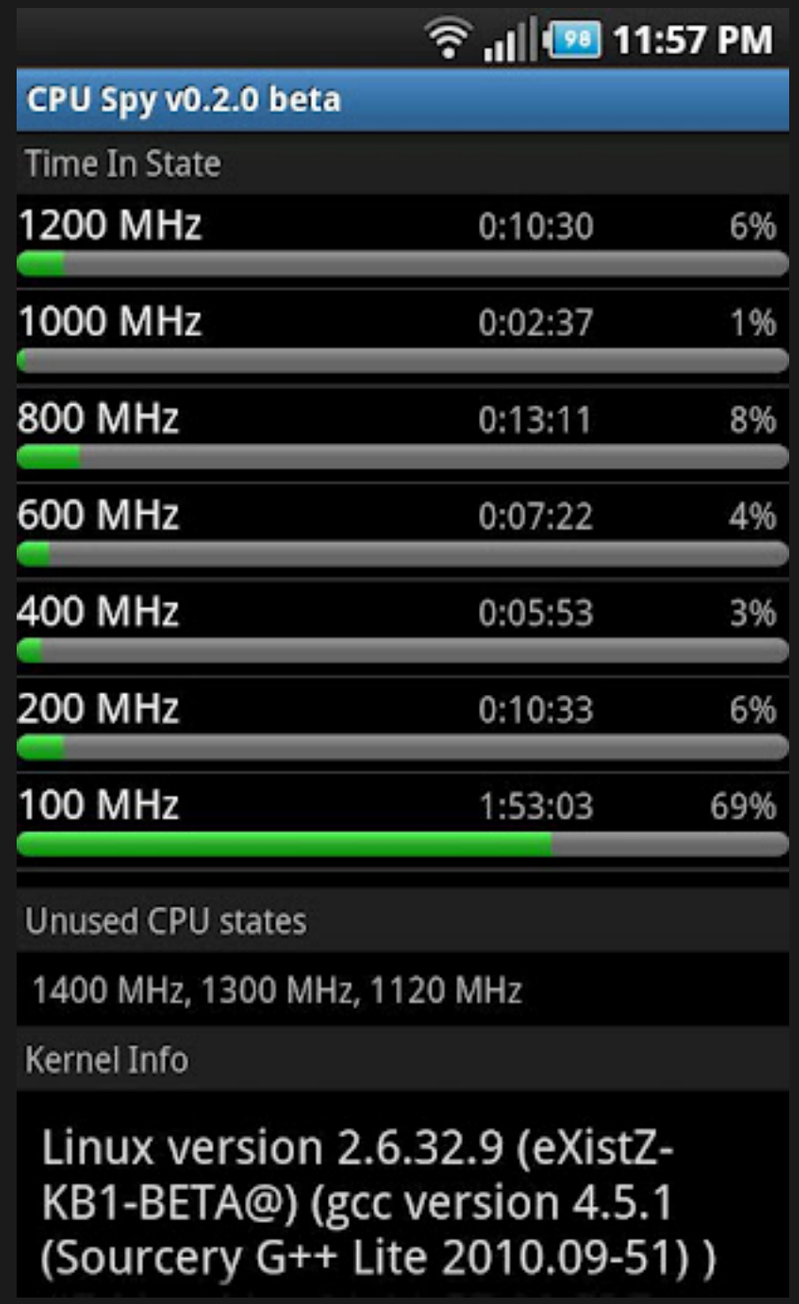
- typically runs on various frequencies
- frequency transition controlled by governor policy installed
- same concept as on linux based PCs, but typically more aggressive
- system drops down to ‘deep sleep’ as much as it can
- frequencies are scaled based of usage



(C) CPUspy

cpu

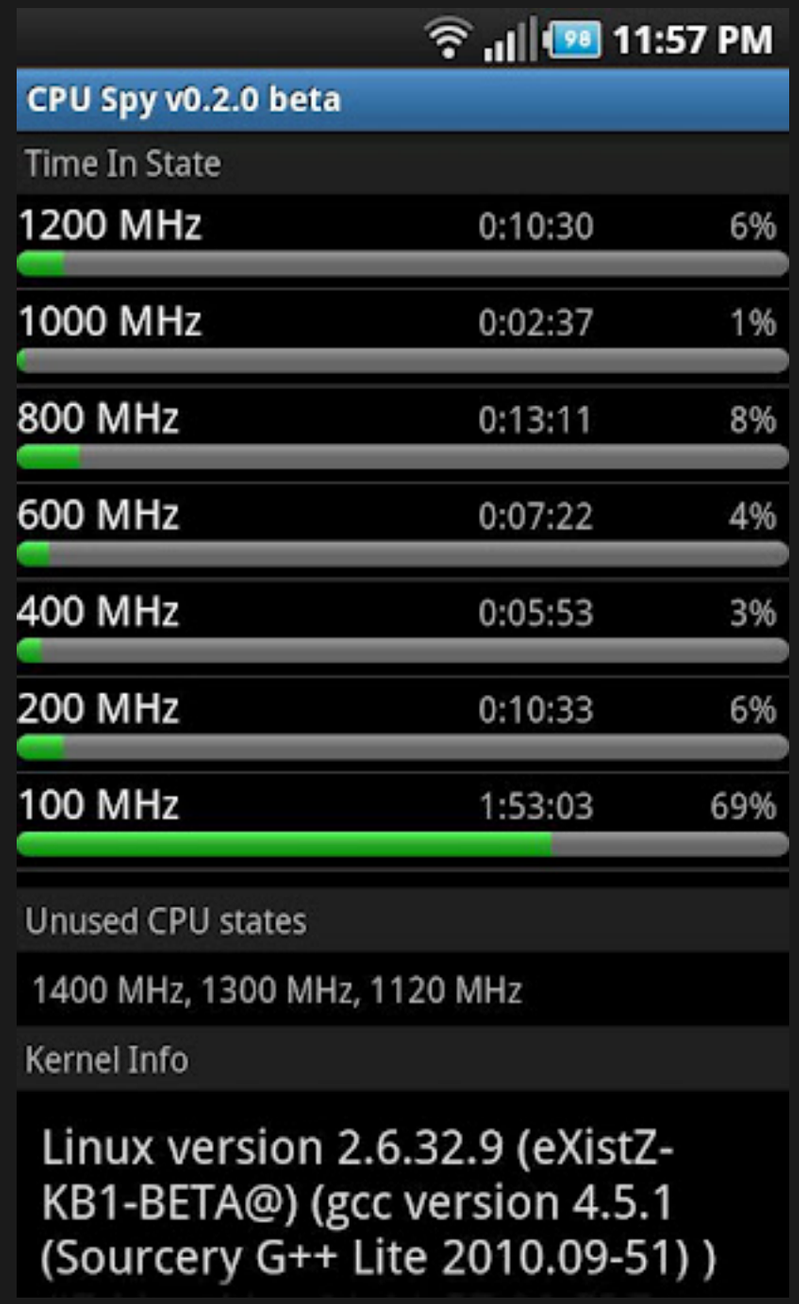
- typically runs on various frequencies
- frequency transition controlled by governor policy installed
- same concept as on linux based PCs, but typically more aggressive
- system drops down to ‘deep sleep’ as much as it can
- frequencies are scaled based of usage



(C) CPUspy

cpu

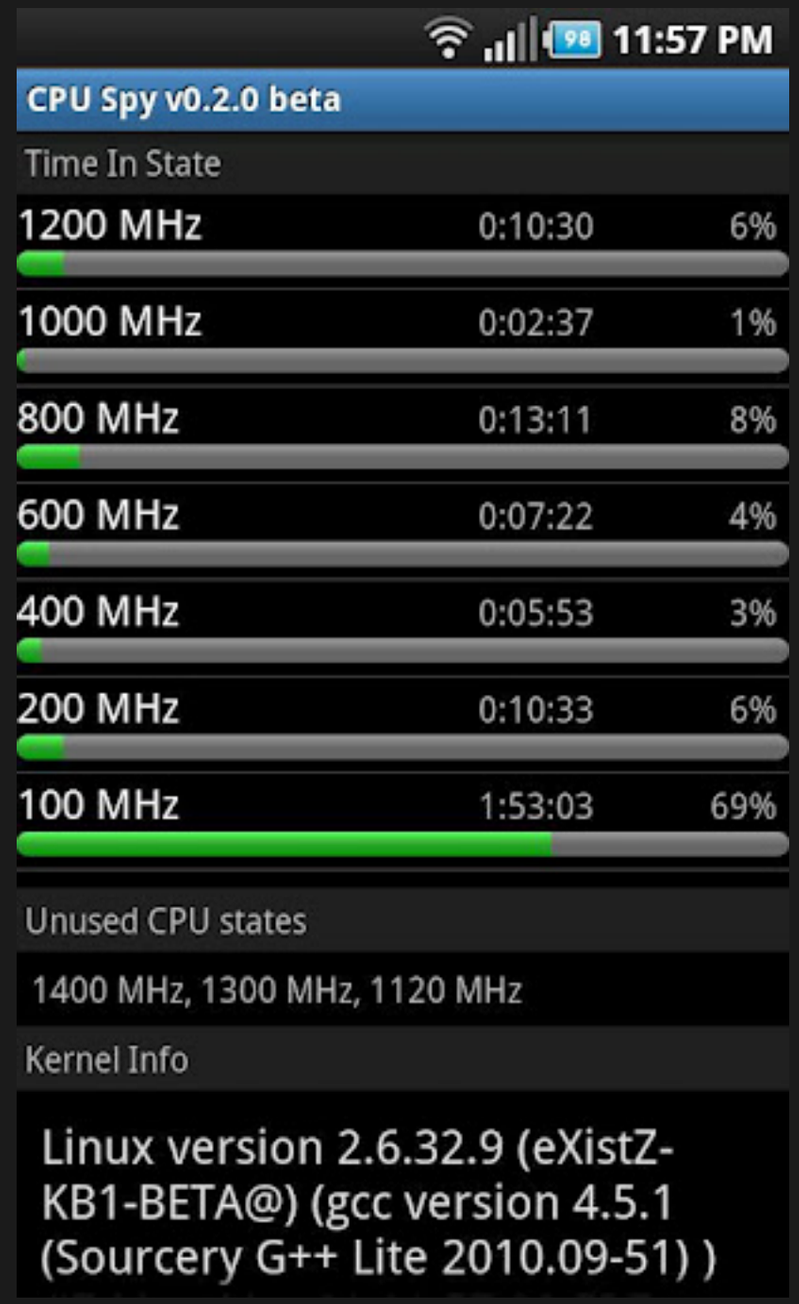
- typically runs on various frequencies
- frequency transition controlled by governor policy installed
- same concept as on linux based PCs, but typically more aggressive
- system drops down to ‘deep sleep’ as much as it can
- frequencies are scaled based of usage



(C) CPUspy

cpu

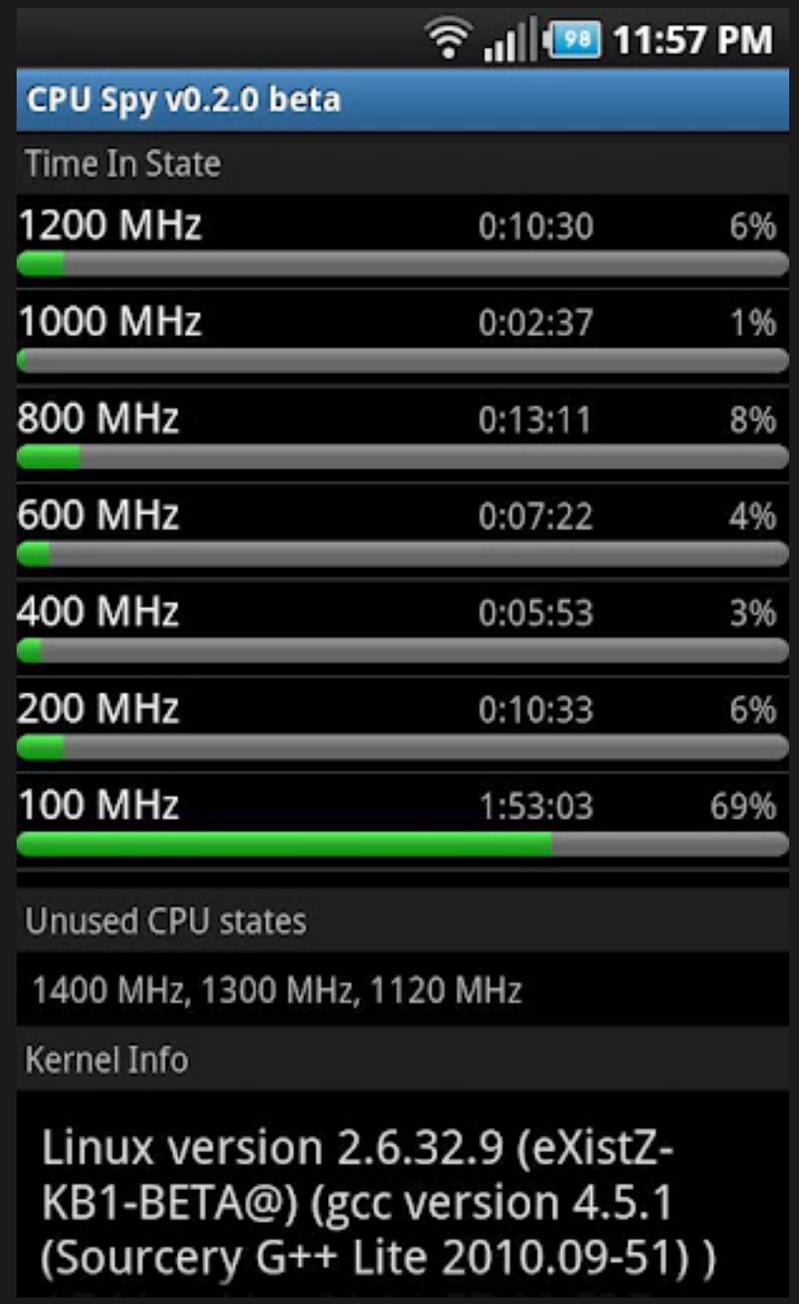
- typically runs on various frequencies
- frequency transition controlled by governor policy installed
- same concept as on linux based PCs, but typically more aggressive
- system drops down to ‘deep sleep’ as much as it can
- frequencies are scaled based of usage



(C) CPUspy

cpu

- typically runs on various frequencies
- frequency transition controlled by governor policy installed
- same concept as on linux based PCs, but typically more aggressive
- system drops down to ‘deep sleep’ as much as it can
- frequencies are scaled based of usage



(C) CPUspy

best practices for cpu

- wakelocks are your friend, but your users worst nightmare!
- use them only if really needed and ensure they are removed as soon as possible
- use *'android:keepScreenOn'* property in your manifest instead of doing it programmatically
- if possible, spread out your computationally intensive job

best practices for cpu

- wakelocks are your friend, but your users worst nightmare!
- use them only if really needed and ensure they are removed as soon as possible
- use *'android:keepScreenOn'* property in your manifest instead of doing it programmatically
- if possible, spread out your computationally intensive job

best practices for cpu

- wakelocks are your friend, but your users worst nightmare!
- use them only if really needed and ensure they are removed as soon as possible
- use *'android:keepScreenOn'* property in your manifest instead of doing it programmatically
- if possible, spread out your computationally intensive job

best practices for cpu

- wakelocks are your friend, but your users worst nightmare!
- use them only if really needed and ensure they are removed as soon as possible
- use *'android:keepScreenOn'* property in your manifest instead of doing it programmatically
- if possible, spread out your computationally intensive job

best practices for cpu

- gpu typically consumes more power than cpu
  - avoid floating point math where possible
- use gpu for data-parallel tasks like video/image processing
- use algorithms that consume less CPU cycle
  - O(n log n) vs O(n 2) algorithms

goal - keep the freq to the lowest level or reduce the number of cycles

best practices for cpu

- gpu typically consumes more power than cpu
  - avoid floating point math where possible
- use gpu for data-parallel tasks like video/image processing
- use algorithms that consume less CPU cycle
  - O(n log n) vs O(n 2) algorithms

goal - keep the freq to the lowest level or reduce the number of cycles

best practices for cpu

- gpu typically consumes more power than cpu
  - avoid floating point math where possible
- use gpu for data-parallel tasks like video/image processing
- use algorithms that consume less CPU cycle
  - O(n log n) vs O(n 2) algorithms

goal - keep the freq to the lowest level or reduce the number of cycles

best practices (others)

- gps is another very expensive component
  - avoid fine grained locations if possible
- use power efficient SDKs
  - eg: Location SDK from SkyhookWireless

best practices (others)

- gps is another very expensive component
  - avoid fine grained locations if possible
- use power efficient SDKs
  - eg: Location SDK from SkyhookWireless

SKYHOOK

references

1. http://developer.android.com/training/efficient-downloads/index.html
2. http://developer.android.com/training/monitoring-device-state/index.html
3. http://dl.google.com/io/2009/pres/W_0300_CodingforLife-BatteryLifeThatIs.pdf
4. http://people.cs.umass.edu/~arun/papers/TailEnder.pdf
5. http://chipdesignmag.com/lpd/absolute-power/2012/05/10/4g-lte-vs-3g/
6. http://thisweekinbatteries.blogspot.in/
7. http://www.ruf.rice.edu/~mobile/publications/dong10chameleon-demo.pdf

little eye labs

thank you

this presentation has been shared @

http://www.slideshare.net/littleeye/power-optimization-for-android-developers

http://www.littleeye.co/blog

kumar@littleeye.co