Energy Efficient Applications for Low Powered Devices

Anjali Singhal
M.Tech. II
Computer Science and Engineering
Indian Institute of Technology Bombay

3rd Mar 2013

AAKASH
tablet for every indian
NME-ICT(MHRD),AAKASH PROJECT, IIT BOMBAY

Indian Institute of Technology Bombay

सत्यमेव जयते
The National Mission on Education through ICT (NME-ICT)

Functionality

http://potofthots.com/2012/04/miss-the-good-old-nokia-3310.html

Functionality

http://potofthots.com/2012/04/miss-the-good-old-nokia-3310.html

http://www.mainstreet.com/slideshow/smart-spending/technology/10-best-apps-2011

Functionality

http://potofthots.com/2012/04/miss-the-good-old-nokia-3310.html

+

http://www.mainstreet.com/slideshow/smart-spending/technology/10-best-apps-2011

=

http://img329.imageshack.us/img329/1218/iphoneallwhiteym1.png

http://potofthots.com/2012/04/miss-the-good-old-nokia-3310.html

More Apps ---> More Power Consumption

http://potofthots.com/2012/04/miss-the-good-old-nokia-3310.html

More Apps ---> More Power Consumption

Developers of these Apps generally care about functionality and not power consumption of their apps!

↓

Battery of these phones drains very quickly, hence end users suffer.

http://potofthots.com/2012/04/miss-the-good-old-nokia-3310.html

More Apps ---> More Power Consumption

Each developer can make an impact by improving energy consumption of his own application

Division of Power consumption

- Most of power consumption can be attributed to the GSM module, CPU and display, including the LCD panel and touchscreen, the graphics accelerator/driver, and the backlight[2]

Division of Power consumption

- The most effective power management approach is to shut down the unused components.

Division of Power consumption

- It has also been found that free-advertisement module consumes 65–75% of the total energy[7].

Agenda

- Energy Bugs
- Optimizations
  - Power-Aware Application Design
  - Battery Virtualization
  - Network Applications
- Other Optimizations
- Conclusion

Agenda

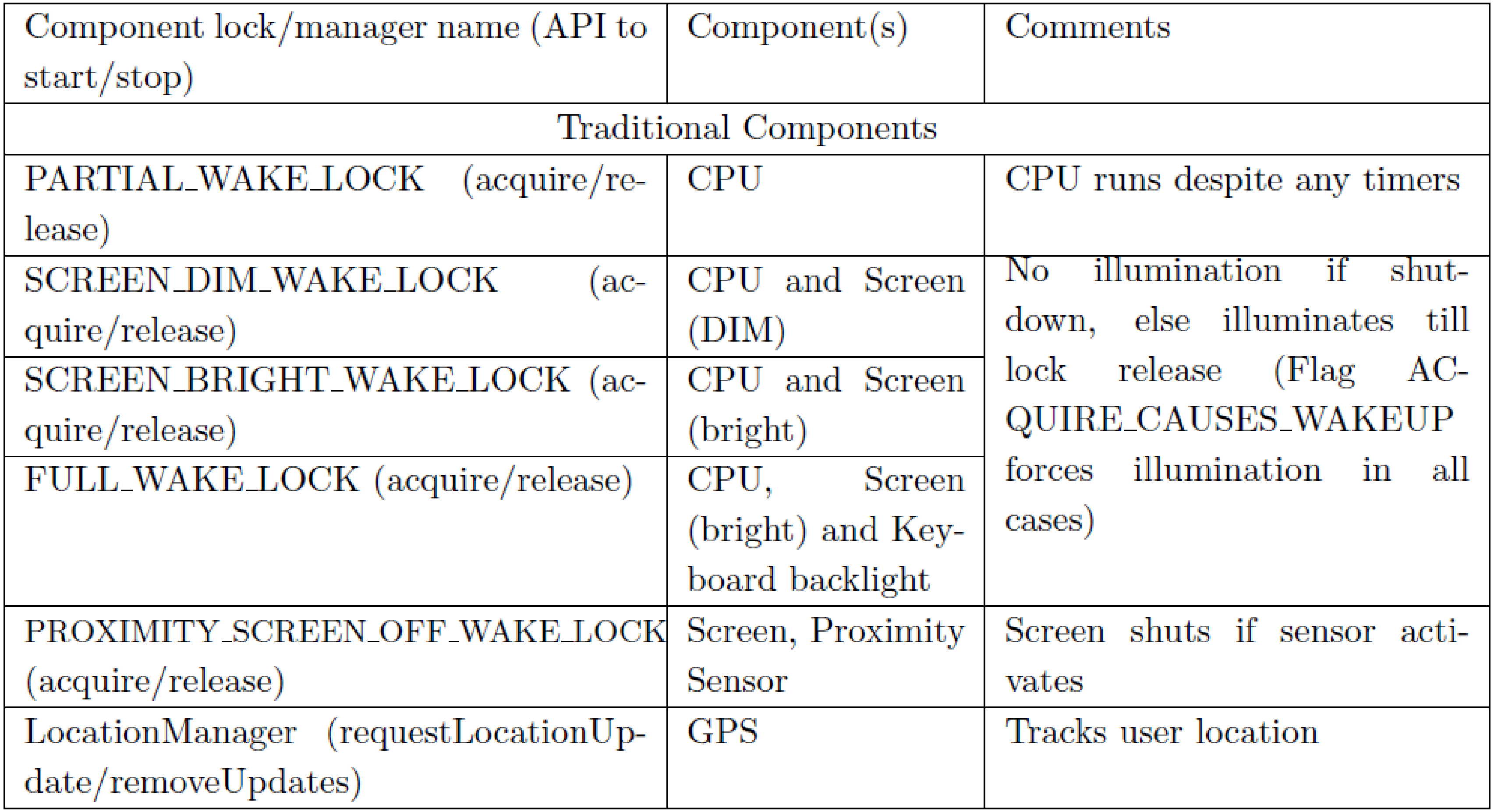
- Energy Bugs
- Optimizations
  - Power-Aware Application Design
  - Battery Virtualization
  - Network Applications
- Other Optimizations
- Conclusion

Energy Bug

- Every component (CPU, WiFi, NIC, 3G, memory, screen, GPS, camera) remains in sleep state, until its waken up explicitly.

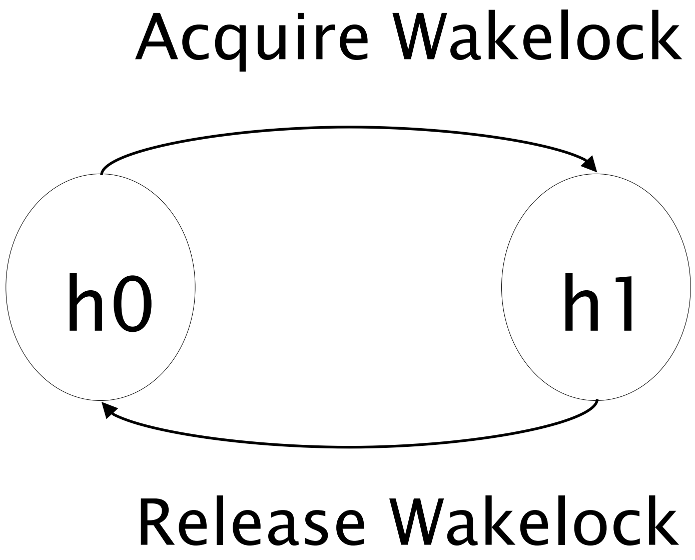
- Power Management in Android
- A wakelock is an instance of *PowerManager.Wakelock* class with one of four options to switch on/ off components related to that option.

Power Management in Android



Component lock/manager name (API to start/stop)	Component(s)	Comments
Traditional Components		
PARTIAL_WAKE_LOCK (acquire/release)	CPU	CPU runs despite any timers
SCREEN_DIM_WAKE_LOCK (acquire/release)	CPU and Screen (DIM)	No illumination if shutdown, else illuminates till lock release (Flag ACQUIRE_CAUSES_WAKEUP forces illumination in all cases)
SCREEN_BRIGHT_WAKE_LOCK (acquire/release)	CPU and Screen (bright)	
FULL_WAKE_LOCK (acquire/release)	CPU, Screen (bright) and Keyboard backlight	
PROXIMITY_SCREEN_OFF_WAKE_LOCK (acquire/release)	Screen, Proximity Sensor	Screen shuts if sensor activates
LocationManager (requestLocationUpdate/removeUpdates)	GPS	Tracks user location

Power Management in Android



h0:low power state

h1:high power state

Usage of Wakelock

- PowerManager.WakeLock wl = pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK);
- wl.acquire(); //CPU should not go to sleep
- // ur code
- wl.release(); //CPU is free to sleep

Energy Bug

- Energy bug or ebug[8] is defined as an error in system due to which there is unexpected amount of energy drain.

- Applications or OS continue to provide normal functionality but with huge energy drain.

- No Sleep Bug

- Looping Bug

Activity

- An activity is started in *onCreate* and destroyed in *onDestroy* callbacks respectively.

- It is paused when it is not in foreground, but visible, and when it goes to background; it stops, but is not destroyed.

- So, if the developer releases the wakelock in *onDestroy* event, but not *onPause*, there exists no sleep code path.

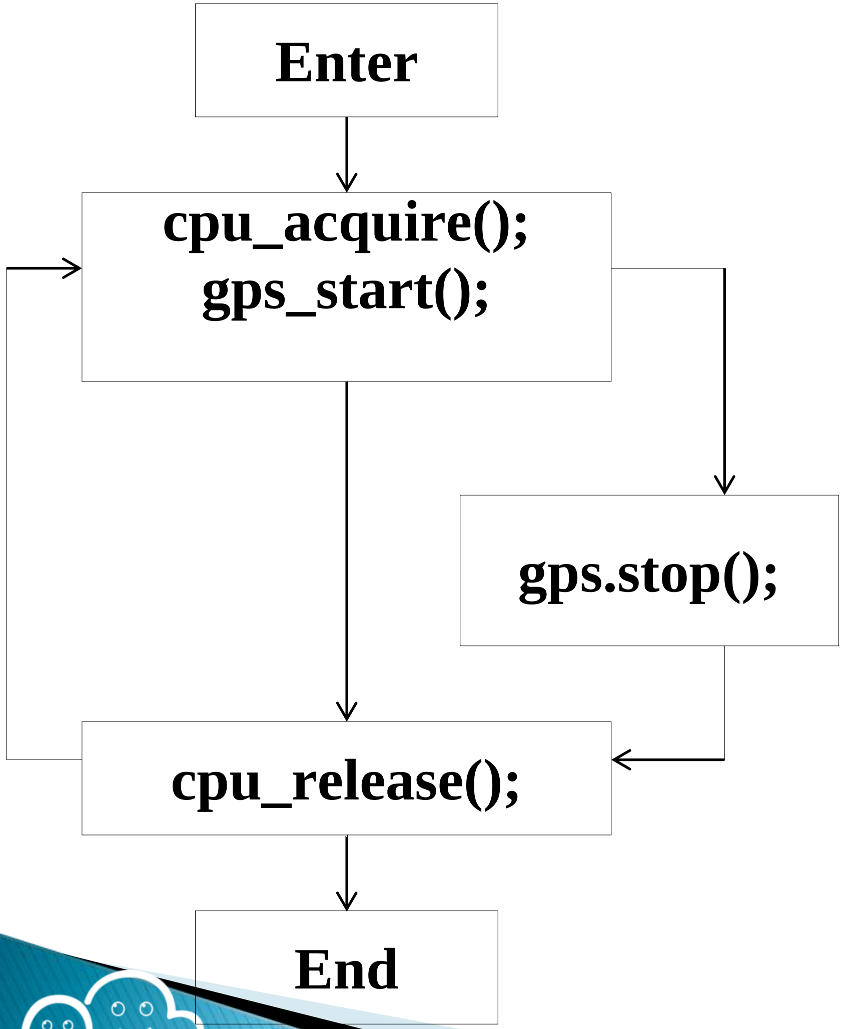
Releasing wakelocks

- Service : At the end of *onStartCommand*, *onUnbind*, and *onHandleIntent* callbacks the desired task should be completed and wakelock should be released.

- Broadcast Receivers : At the end of *onReceive* callback.

- By Static data flow analysis, no sleep code paths can be found.

No Sleep Bug



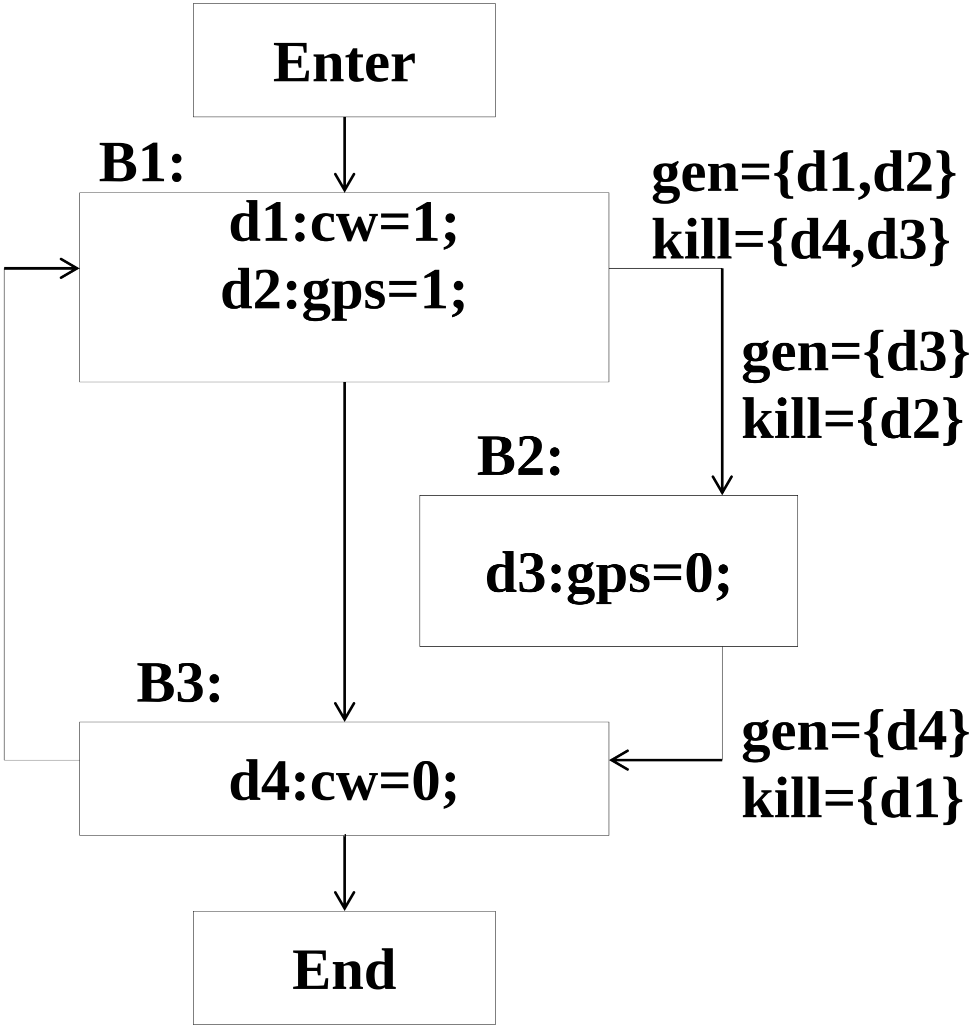
Enter

cpu_acquire();
gps_start();

gps.stop();

cpu_release();

End



Enter

B1:

d1:cw=1;
d2:gps=1;

gen={d1,d2}
kill={d4,d3}

gen={d3}
kill={d2}

B2:

d3:gps=0;

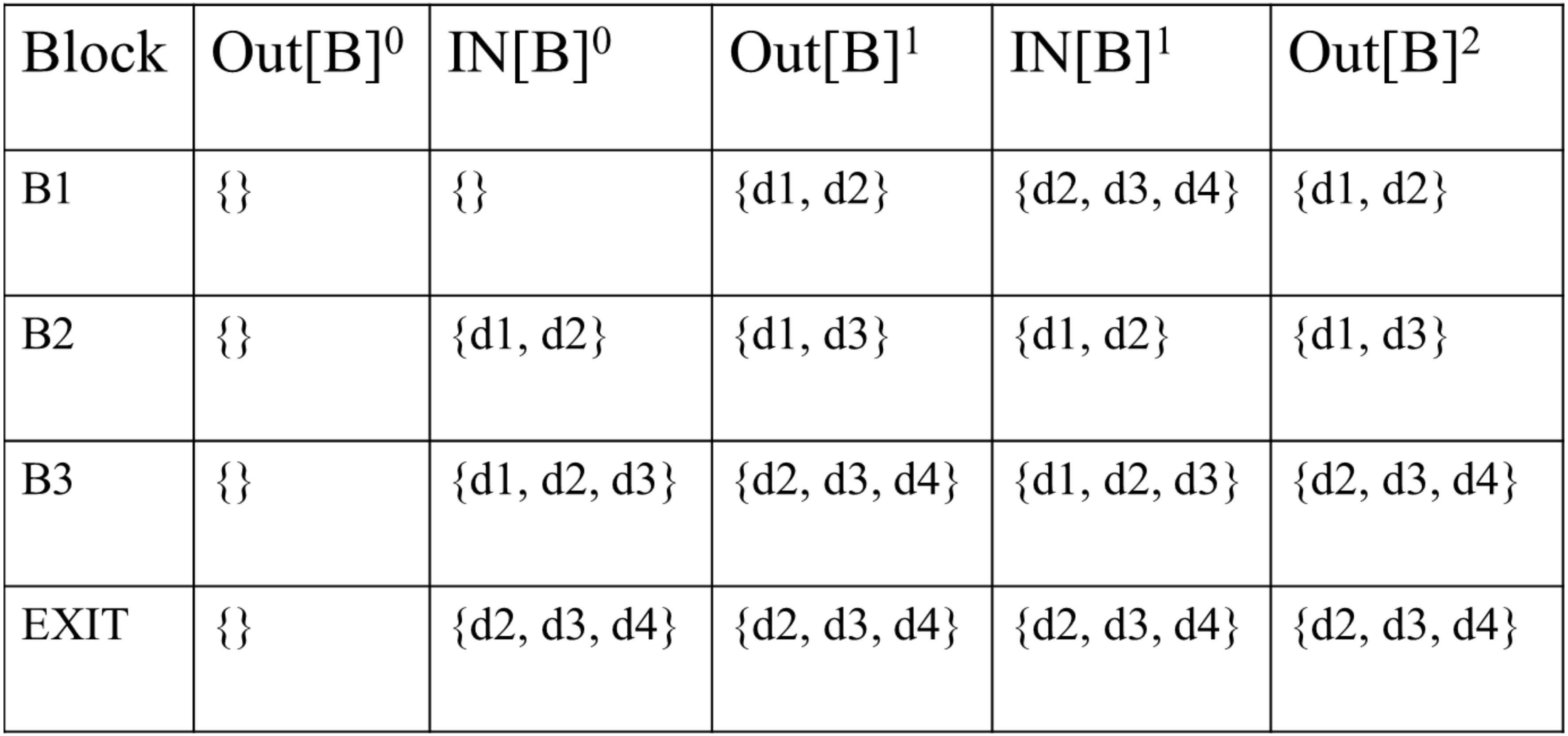
B3:

d4:cw=0;

gen={d4}
kill={d1}

End

No Sleep Bug Analysis



Block	$\text{Out}[B]^0$	$\text{IN}[B]^0$	$\text{Out}[B]^1$	$\text{IN}[B]^1$	$\text{Out}[B]^2$
B1	{}	{}	{d1, d2}	{d2, d3, d4}	{d1, d2}
B2	{}	{d1, d2}	{d1, d3}	{d1, d2}	{d1, d3}
B3	{}	{d1, d2, d3}	{d2, d3, d4}	{d1, d2, d3}	{d2, d3, d4}
EXIT	{}	{d2, d3, d4}	{d2, d3, d4}	{d2, d3, d4}	{d2, d3, d4}

Agenda

- Energy Bugs
- Optimizations
  - Power-Aware Application Design
  - Battery Virtualization
  - Network Applications
- Other Optimizations
- Conclusion

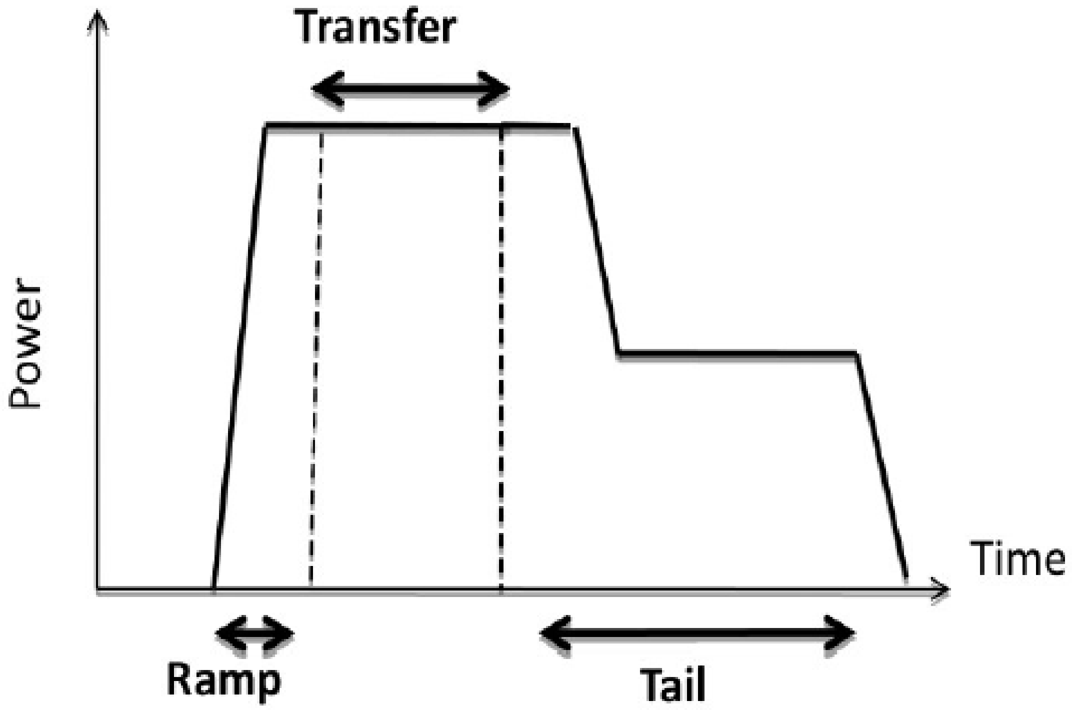
Optimizations

- Power–aware application design[4]:
  - The design choice does not change lines of code, but can impact energy usage. e.g read/write compressed/ uncompressed file.
  - Applications needing continuous but variable workload, may benefit most from processor frequency scaling, etc.

Optimizations

- Battery Virtualization:
  - Having a battery allocation for each application class (navigation, phone, game), will ensure that they have access to a fraction of the battery, as per user's individual policy.
  - It can be enforced as a policy through an Android service which will periodically check for increase in energy consumption over the fraction assigned to that class.

Optimizations in Network Applications



Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network appli- cations. In Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, IMC ’09, pages 280–293, New York, NY, USA, 2009. ACM

Optimizations in Network Applications(2)

- Network devices are major consumers of energy in most of the applications.

- 3G Measurements :

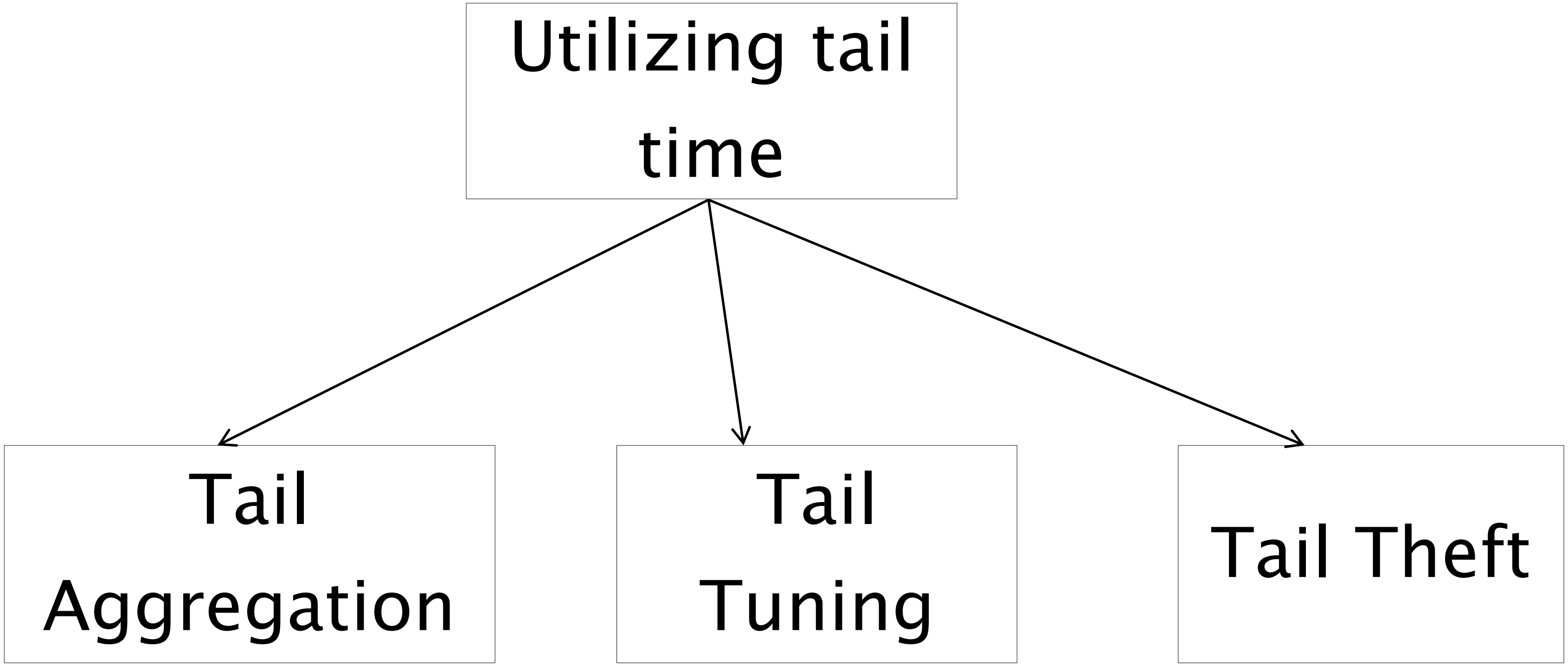
  Tail energy is a significant fraction of the total energy, while ramp energy is significantly small compared to tail energy.

Optimizations in Network Applications(3)

- GSM Measurements : The Tail energy is comparable to the transfer energy; but it is less than the tail energy consumed in 3G.

- WiFi Measurements : The scanning and association energy is comparable or greater than the transfer energy, with high maintenance energy.

Optimizations in Network Applications(4)



Utilizing tail time

Tail Aggregation

Tail Tuning

Tail Theft

Optimizations in Network Applications(5)

- Tail Aggregation[1] :
  - Each request if deferred until its deadline in case of delay/tolerant apps, so that their tail energies overlap.
  - Due to overlap, the inter transfer time decreases, decreasing the time for which the component is in high power state.

Optimizations in Network Applications(6)

- Tail Tuning :
  - If the tail time is reduced, then the number of state promotions may increase a lot, thereby increasing power consumption. Hence, very high prediction accuracy is required.

Optimizations in Network Applications(7)

- Tail Theft[5] :
  - Virtual tail is maintained along with the physical one, and small transmissions are sent in this virtual tail time.
  - If the tail time ends before transmission, then that transmission is cancelled.

Agenda

- Energy Bugs
- Optimizations
  - Power-Aware Application Design
  - Battery Virtualization
  - Network Applications
- Other Optimizations
- Conclusion

Other Optimizations

- After network components, display consumes the highest power.

- The LCD backlight[3] is the dominant power consumer, with the LCD panel and the framebuffer, coming second and third respectively.

- Dynamically changing brightness according to the ambience, and encoding to compress framebuffer.

Agenda

- Energy Bugs
- Optimizations
  - Power-Aware Application Design
  - Battery Virtualization
  - Network Applications
- Other Optimizations
- Conclusion

Conclusion

- We studied energy bugs, their detection, and possible causes.

- We also studied different optimizations that can be done in different applications such as network applications, etc., by properly utilizing tail time, and different enhancements that can be done to them.

References

1. Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, IMC ’09, pages 280–293, New York, NY, USA, 2009. ACM

2. Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In Proceedings of the 2010 USENIX conference on USENIX annual technical conference, USENIXATC’10, pages 21–21, Berkeley, CA, USA, 2010. USENIX Association

3. Inseok Choi, Hojun Shim, and Naehyuck Chang. Low-power color tft lcd display for handheld embedded systems. In Proceedings of the 2002 international symposium on Low power electronics and design, ISLPED ’02, pages 112–117, New York, NY, USA, 2002. ACM

References

4. Aman Kansal and Feng Zhao. Fine-grained energy profiling for power-aware application design. SIGMETRICS Perform. Eval. Rev., 36(2):26–31, August 2008

5. Hao Liu, Yaoxue Zhang, and Yuezhi Zhou. Tailtheft: leveraging the wasted time for saving energy in cellular communications. In Proceedings of the sixth international workshop on MobiArch, MobiArch '11, pages 31–36, New York, NY, USA, 2011. ACM

6. Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. Bootstrapping energy debugging on smartphones: a first look at energy bugs in mobile devices. In Proceedings of the 10th ACM Workshop on Hot Topics in Networks, HotNets-X, pages 5:1–5:6, New York, NY, USA, 2011. ACM

7. Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In Proceedings of the 7th ACM european conference on Computer Systems, EuroSys '12, pages 29–42, New York, NY, USA, 2012. ACM.

References

8. Abhinav Pathak, Abhilash Jindal, Y. Charlie Hu, and Samuel P. Midkiff. What is keeping my phone awake?: characterizing and detecting no-sleep energy bugs in smartphone apps. In Proceedings of the 10th international conference on Mobile systems, applications, and services, MobiSys ’12, pages 267–280, New York, NY, USA, 2012. ACM

9. Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, CODES/ISSS ’10, pages 105–114, New York, NY, USA, 2010. ACM.