Cellular Networks and Mobile Computing
COMS 6998-8, Spring 2012

Instructor: Li Erran Li
(lierranli@cs.columbia.edu)

http://www.cs.columbia.edu/~coms6998-8/

2/20/2012: Ebugs, Power Models, Profiling and Debuging

Announcements

- To obtain physical access to Gateway Lab, contact mjg2203@columbia.edu
- Contact TA Hemin Merchant to provision your iOS devices
- Contact TA Jiawen Sun to get Amazon EC2 credits (one representative from each project team)
- Programming assignment 1 will be due on Monday, Feb 27th

Outline

- The Rise of Ebugs
- Methods of Measuring Power Usage
- Power Models
  - Usage based
  - System call trace based
- Profiling
- Conclusion

The Rise of Energy Bugs

Single Symptom:

Severe, **Unexpected** Battery Drain

Apps Need Not Crash
No Blue Screen Of Death

Common Perception:
Kill some apps to fix

Courtesy: Pathak et al

User Frustration (Dialer App EBug)

Comment 24 by mgil...@gmail.com, Aug 14, 2011

```
This defect is a real P.I.T.A. - I don't want to use my phone as a phone because I
have to restart it every time.  If I forget then it's usually 30-40% battery gone by
the end of the day.
```

Comment 30 by hansheng...@gmail.com, Aug 15, 2011

```
Bring your charger with you and keep it charged!!! That's the only way the phone can last a day.
It's a irritating bug!!!
```

Comment 239 by egork...@gmail.com, Nov 6 (6 days ago)

```
GOOGLE!!!!!!!! DO SOMETHING WITH THIS ISSUE!!! FASTER PLEASE!!!!
```


Courtesy: Pathak et al

Crawling Internet Forums

1. **4 Online Mobile Forums** → **1.2M Posts** **(~400 mobile devices; Several mobile OSes)**

→ (grep "power drain", "battery drain", etc.) → **39K Posts**

→ (K-means) → **1000 Clusters**

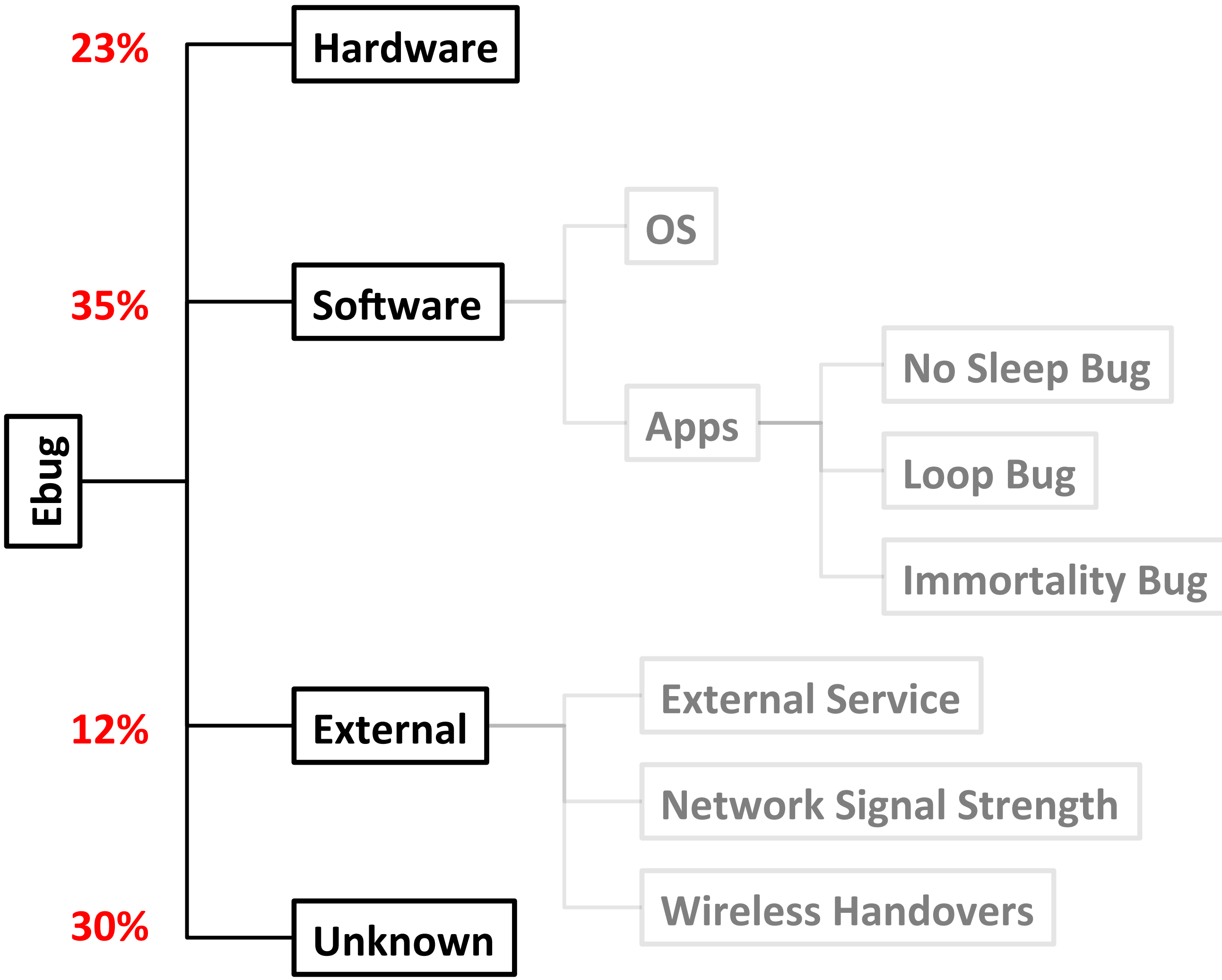
→ **Human** → **EBug Taxonomy**

....

2. **2 Mobile Bugs/Issues Repository** → **Indepth Analysis**

Courtesy: Pathak et al

Ebug Taxonomy



Courtesy: Pathak et al

Hardware EBug

Ebug
- **Hardware** 23%
- Software 35%
- External 12%
- Unknown 30%

Battery

External Hardware

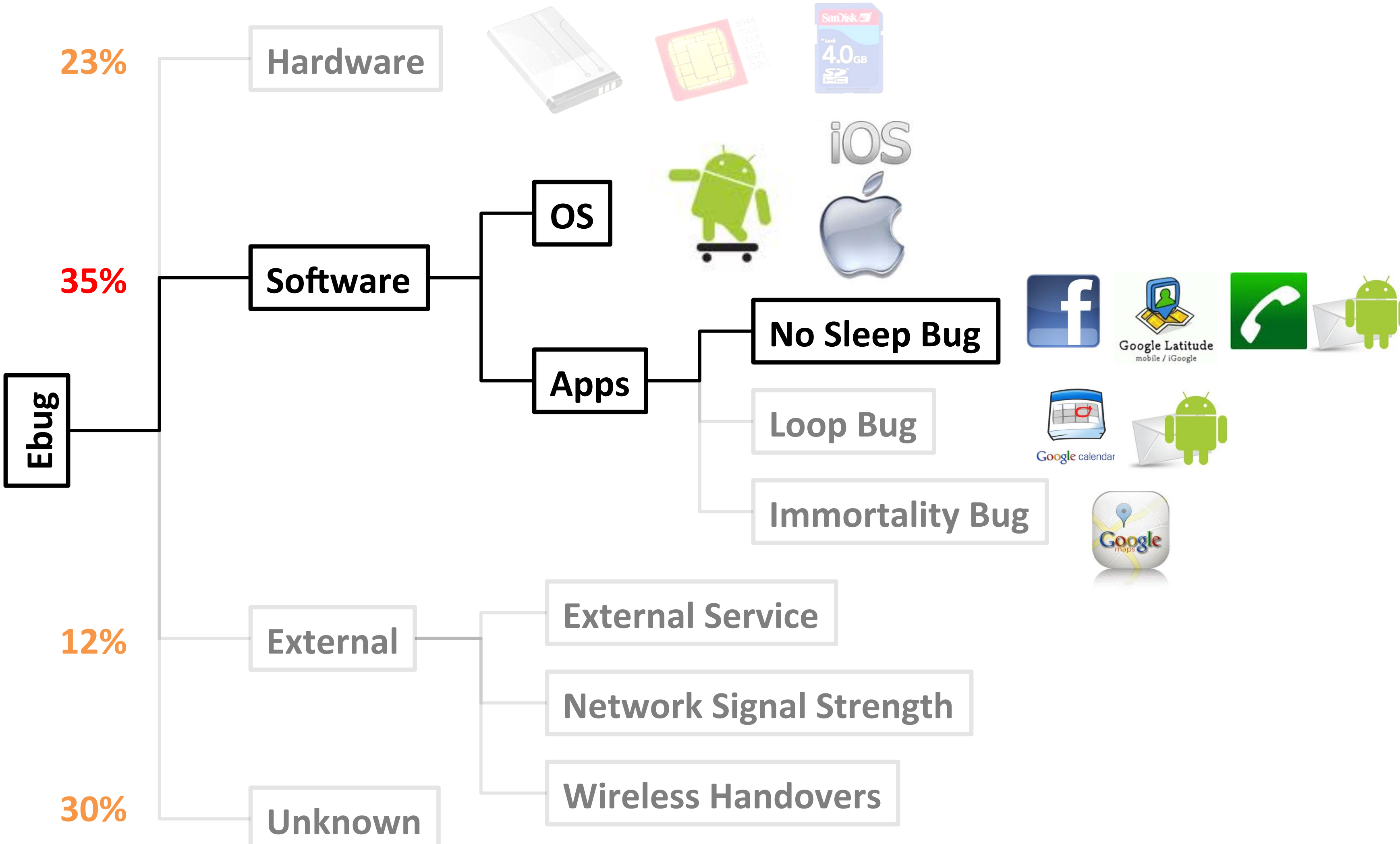
Sim Card

Exterior Hardware Damage

SDCard

Courtesy: Pathak et al

Ebug Taxonomy



Courtesy: Pathak et al

OS Ebugs

Why does OS Leak Energy?

– Hard to infer

– OS Processes

– System Configuration


HOT TOPICS APPLE ANDROID GOOGLE REPUBLIC WIRELESS FACEBOOK ADOBE

iPhone 4S Battery Life Bugs Got You Down?

Battery life on the iPhone 4S: the new 'death grip'?

By **Doug Gross**, CNN
updated 4:17 PM EST, Tue November 1, 2011 | Filed under: Mobile

iPhone battery fix coming 'in a few weeks'

By **Doug Gross**, CNN
updated 11:19 AM EST, Thu November 3, 2011 | Filed under: Mobile

IOS Version: 4.0 – 4.3.3 (5% posts)

2.5% posts

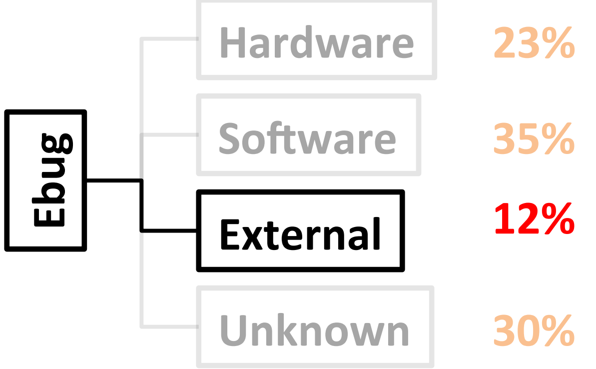
Courtesy: Pathak et al

Apps EBug: No Sleep Bug

- Aggressive Sleeping Policies: Smartphone OSes freeze system after brief inactivity
- Power encumbered Programming: Programmer has to manage sleep/wake cycle of components
- No Sleep Bug: At least one component is kept awake due to mismanagement

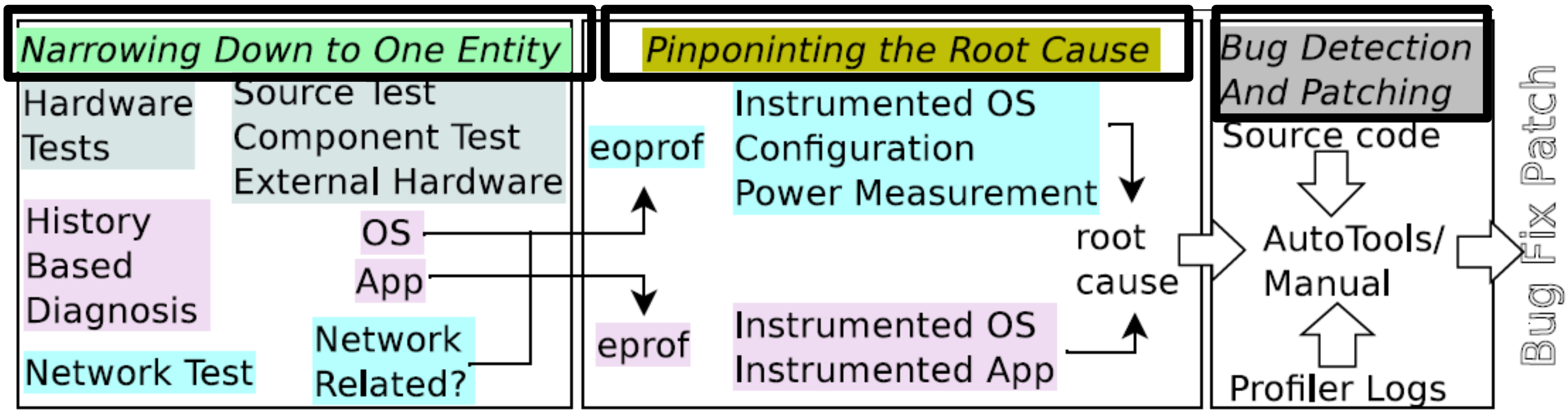
Courtesy: Pathak et al

External Conditions



- External Services (<1%)
- Network Signal Strength (11%)
- Wireless Handovers (<1%)

EDB: Energy Debugging Framework



Courtesy: Pathak et al

Mobile Programming EcoSystem: The EBug Blame Game

Network Operators

App Developers

Hardware Manufacturers

Framework Developers

Firmware/OEM Developers

Kernel Developers

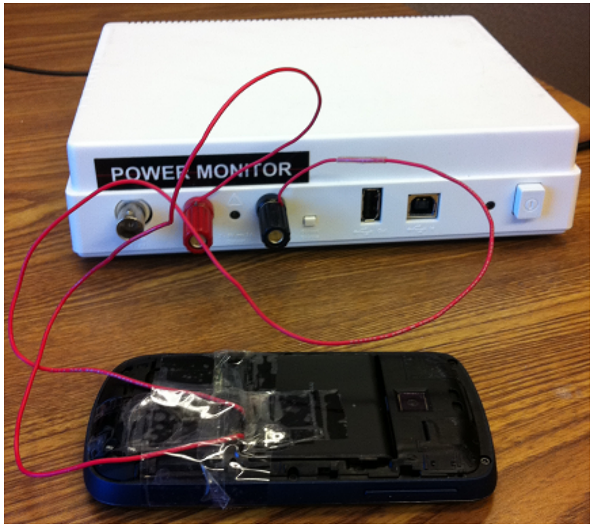
Courtesy: Pathak et al

Outline

- The Rise of Ebugs
- Methods of Measuring Power Usage
- Power Models
  - Usage based
  - System call trace based
- Profiling
- Conclusion

Measuring Power Usage

- Approach 1: Use power meter (offline)
  - Buy an expensive equipment ($770)
  - Problems:
    - Only reports entire device energy consumption



- Approach 2 : Use built-in battery sensor (online)

iOS Battery API

- Use `UIDevice` class to obtain information and notifications about
  - charging state (property `batteryState`)
  - charging level (property `batteryLevel`)

```
[[UIDevice currentDevice] setBatteryMonitoringEnabled:YES];
  NSArray *batteryStatus = [NSArray arrayWithObjects:
                              @"Battery status is unknown.",
                              @"Battery is in use (discharging).",
                              @"Battery is charging.",
                              @"Battery is fully charged.", nil];
  if ([[UIDevice currentDevice] batteryState] == UIDeviceBatteryStateUnknown)
      NSLog(@"%@", [batteryStatus objectAtIndex:0]);
  else
  {
      NSString *msg = [NSString stringWithFormat:
                           @"Battery charge level: %0.2f%%\n%@",
                          [[UIDevice currentDevice] batteryLevel] * 100,
                          [batteryStatus objectAtIndex:[[UIDevice currentDevice]
                                    batteryState]] ];
      NSLog(@"%@", msg);
  }
```

Android Battery API

- Sample updates stored in files:
  - Current: /sys/class/power_supply/battery/batt_chg_current
  - Voltage: /sys/class/power_supply/battery/batt_vol
  - Capacity: /sys/class/power_supply/battery/capacity

```
File fcur = new File("/sys/class/power_supply/
battery/batt_chg_current");
if (fcur.exists())
    …
```

- File names are vendor dependent
- [Access using Android Debug Bridge (adb)]
  - <sdk>platform-tools
  - Command: adb shell

Outline

- The Rise of Ebugs
- Methods of Measuring Power Usage
- Power Models
  - Usage based
  - System call trace based
- Profiling
- Conclusion

Smartphone is Energy Constrained

- Energy: One of the most critical issues in smartphones
  - Limited battery lifetime

- Battery energy density only doubled in last 15 yrs

- Smartphone capability has increased drastically
  - Multiple Components: GPS, 3G, retina display, ...

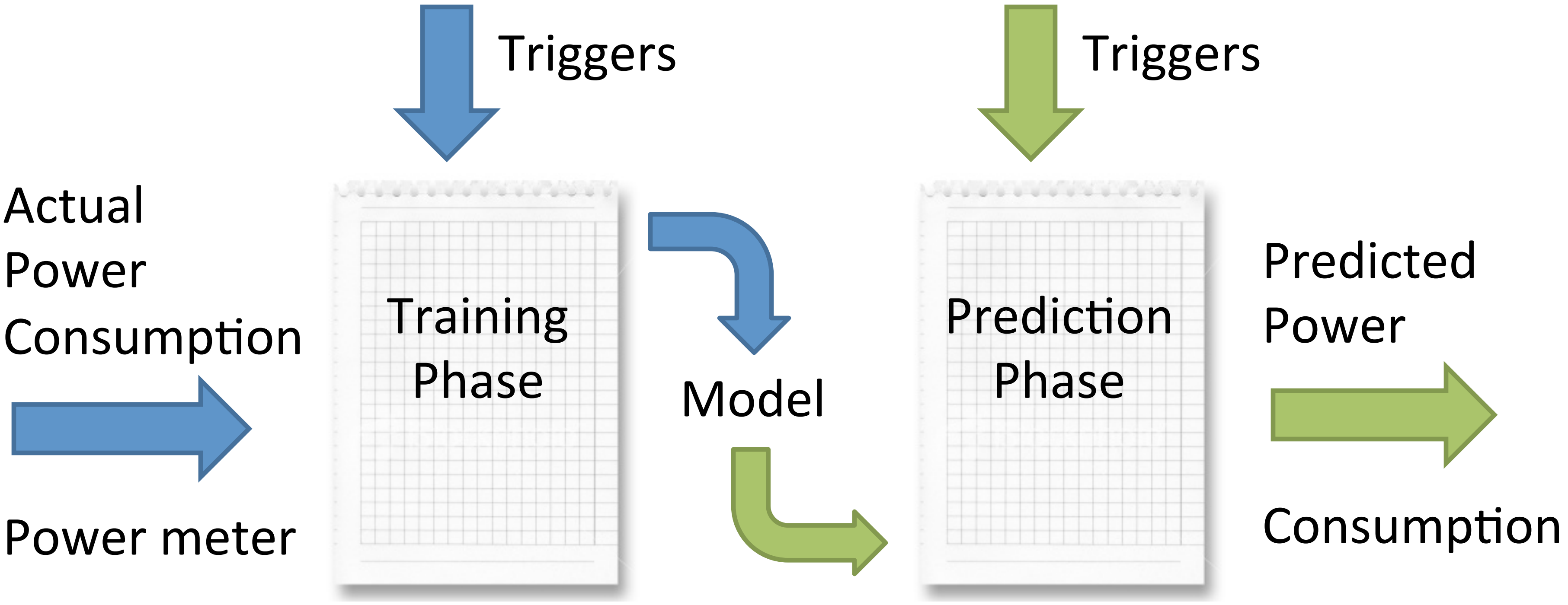
Courtesy: Pathak et al

Towards Understanding Energy Drain

- Key Question: Where is energy being spent?
  - Which component/process/thread/function(?)

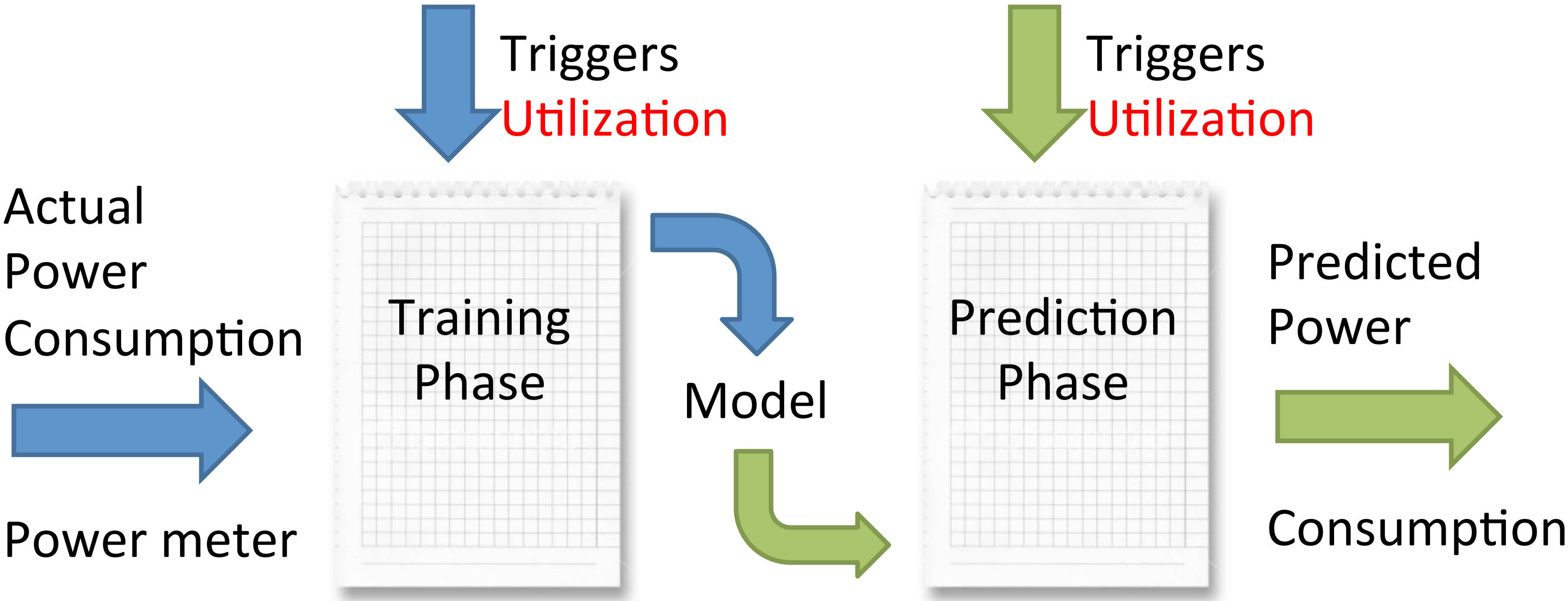
Courtesy: Pathak et al

Generic Power Modeling



Courtesy: Pathak et al

Smartphone Power Modeling: Utilization Based (1/3)



Linear Regression (LR) and Superimposition

$$\text{Model} = (\text{Util}_{Net}) * E_{Net} + (\text{Util}_{CPU}) * E_{CPU} + (\text{Util}_{Disk}) * E_{Disk}$$

Courtesy: Pathak et al

Smartphone Power Modeling: Utilization Based (2/3)

- PowerTutor model

$$(\beta_{uh} \times freq_h + \beta_{ul} \times freq_l) \times util + \beta_{CPU} \times CPU\_on + \beta_{br} \times brightness + \beta_{Gon} \times GPS\_on + \beta_{Gsl} \times GPS\_sl + \beta_{Wi\text{-}Fi\_l} \times Wi\text{-}Fi_l + \beta_{Wi\text{-}Fi\_h} \times Wi\text{-}Fi_h + \beta_{3G\_idle} \times 3G_{idle} + \beta_{3G\_FACH} \times 3G_{FACH} + \beta_{3G\_DCH} \times 3G_{DCH}$$

β : power coefficient.
util, brightness and etc.: system variables.

- Sesame paper has two optimizations: model molding, principle component analysis (PCA)

Smartphone Power Modeling: Utilization Based (3/3)

$\text{Model} = (\text{Util}_{Net})^{*}\ E_{Net} + (\text{Util}_{CPU})^{*}\ E_{CPU} + (\text{Util}_{Disk})^{*}\ E_{Disk}$

Fundamental (yet intuitive) assumption

(Only active) Utilization => power consumption

Second assumption

Energy scales linearly with amount of work

Third assumption

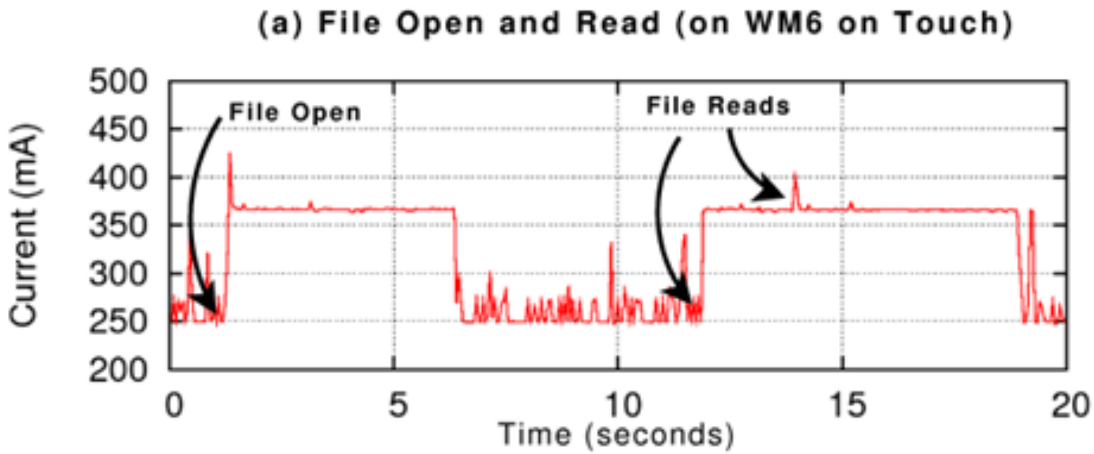
Components power consumption add linearly

Desired Feature

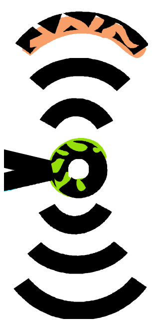
Which process/thread/function? Hard to correlate

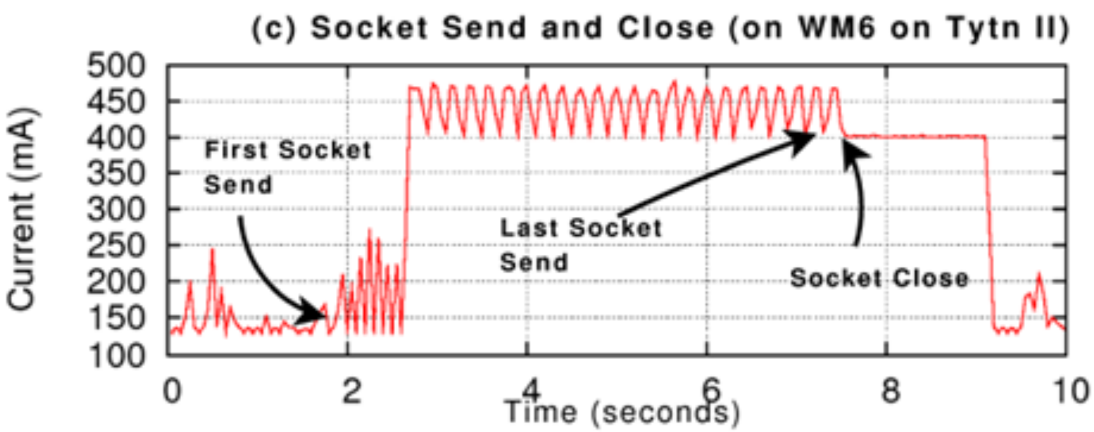
Courtesy: Pathak et al

(Only active) Utilization => Power Consumption



File open/delete/close/create change power state



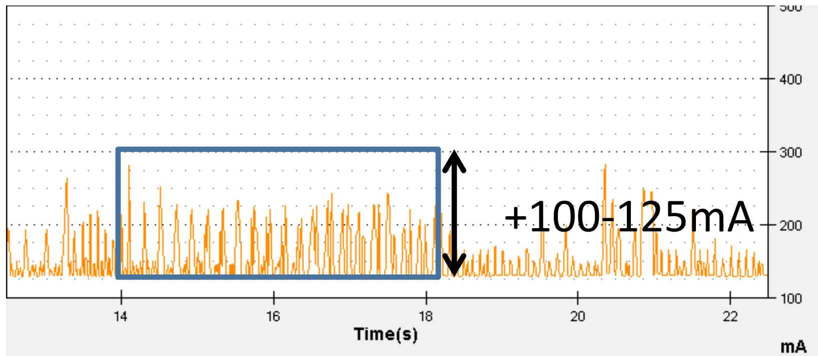


Several components have tail states (3G, disk, wifi, gps)

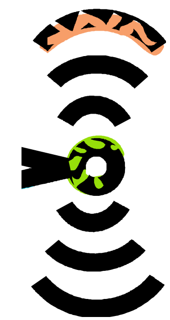
Courtesy: Pathak et al

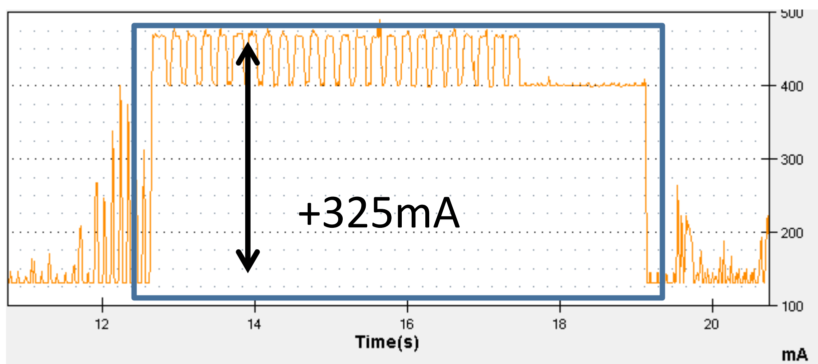
Energy scales linearly with amount of work

WM6.5 on Tytn II



(1) Send packets @ < 50pkts/s



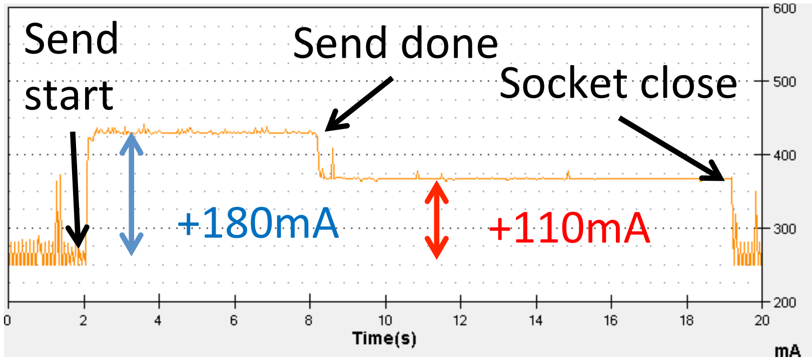


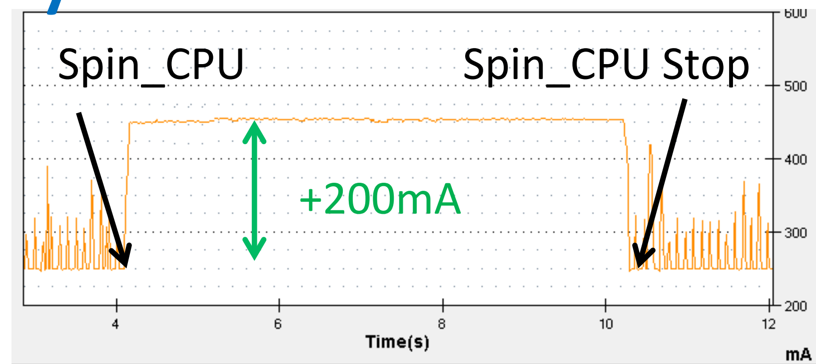
(2) Send packets @ > 50pkts/s

Courtesy: Pathak et al

Components power consumption add linearly

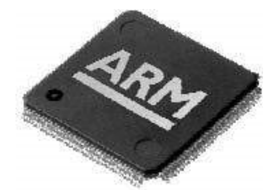
WM6.5 on HTC Touch



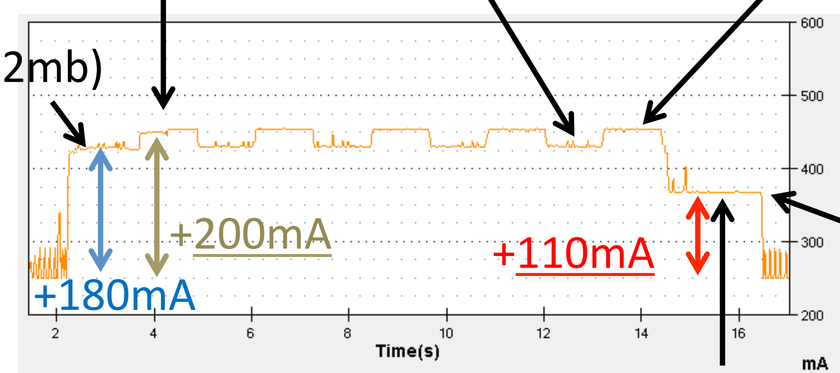


(1) Send(10mb);
sleep();
Socket.close();

(2) Spin_CPU(10M);



(3)
```
for (i in 1 to 5){
    Send(2mb);
    Spin_CPU(2M);
}
Sleep();
Socket.close();
```



Courtesy: Pathak et al

What have we learnt so far?

Simple (state-of-art) energy modeling assumptions are wrong

There exits a notion of power states

What have we hinted so far?

Device drivers have intelligent power control rules

System calls play a role in power consumption

Challenges in fine-grained power modeling?

Device drivers are closed source (no code/no information)

Courtesy: Pathak et al

System Calls As Power Triggers

Key observation: System call is the interface through which an application communicates with the underlying system (hardware) and outside world (Internet, GPS, etc.)

Key Idea: Use System Calls as triggers in power modeling

Advantages:

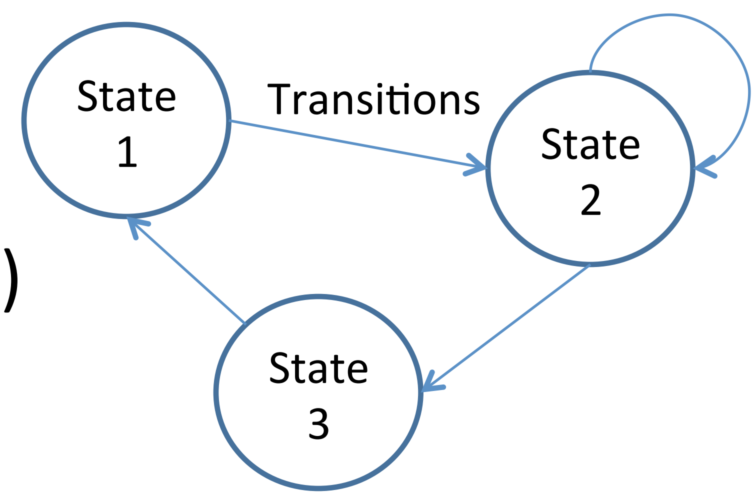
- Encapsulates utilization based triggers
  - Parameters of system calls
- Captures power behavior of ones that do not necessarily imply utilization
- Can be traced back to process, thread, function
  - Eases energy accounting

Courtesy: Pathak et al

Finite-State-Machine (FSM) as Power Model Representation

We Use Finite-State-Machine (FSM)

- Nodes: Power states
  - Base State: No activity on phone
  - Productive state: Actual utilization
  - Tail state: No-useful work
- Edges: Transition rules
  - System calls (start/completion)
  - Workload (Ex: 50 pkts/sec)
  - Timeout



Courtesy: Pathak et al

FSM Power Model Construction

- Systematic 'Brute Force' Approach
  - Step 1 : Model Single System Call
  - Step 2 : Model Multiple System Calls for Same Component
  - Step 3 : Model Multiple Components (Entire Phone)
- Requires domain knowledge
  - Semantics of system calls

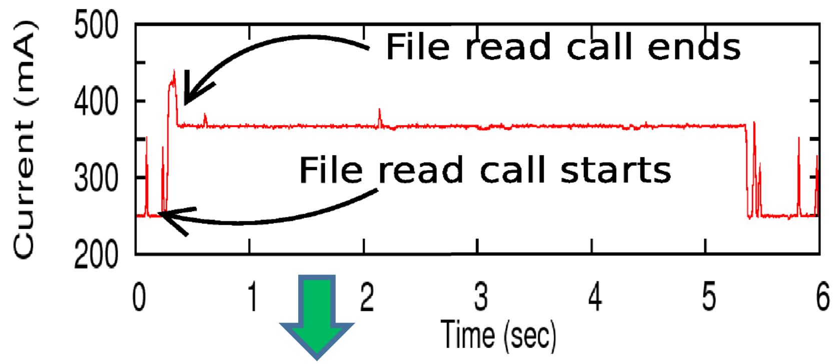
Courtesy: Pathak et al

Step 1: Single System Call FSM

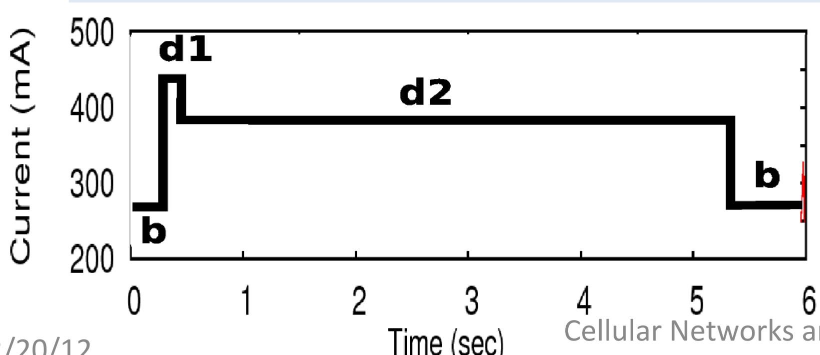
WM6.5 on HTC Touch

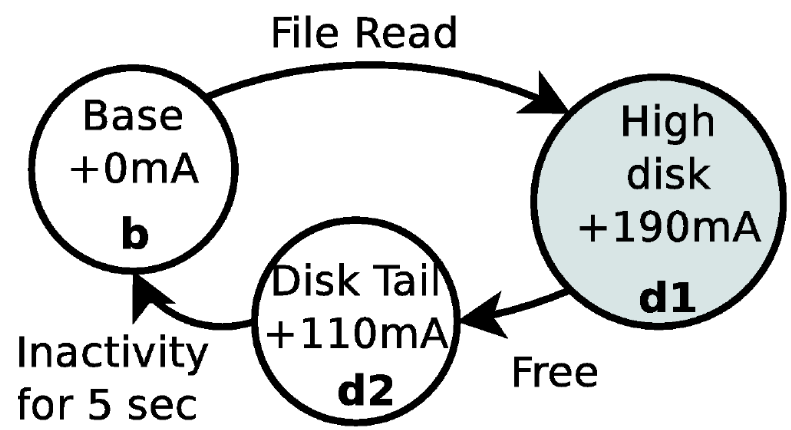
System call: **read** (fd, buf, size);

Measured power consumption + system calls (trigger)



Modeled power consumption





FSM

Courtesy: Pathak et al

Step 2: Modeling Multiple System Calls of Same Component

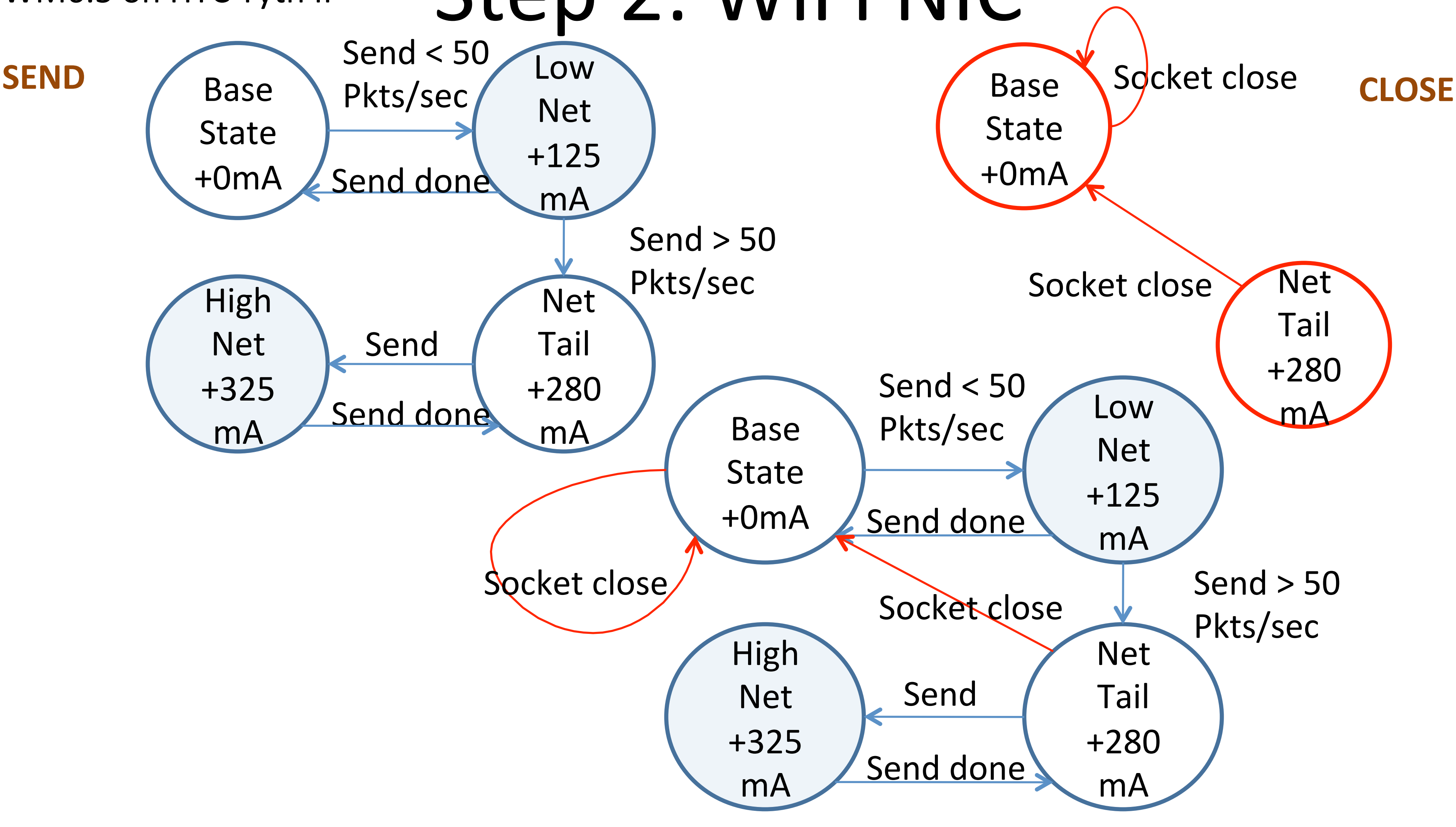
- Observation: A component can only have a small finite number of power states

- Methodology
  - Identify and merge similar power states
  - Obey programming order
  - Model concurrent system calls

 Courtesy: Pathak et al

Step 2: WiFi NIC

WM6.5 on HTC Tytn II



Courtesy: Pathak et al

Step 3: Modeling Multiple Components

- Observation: Different components may interact with each other's power consumption

- Methodology
  - Try to reach different combination of states
  - Construct new states and transitions in FSM

 Courtesy: Pathak et al

Implementation

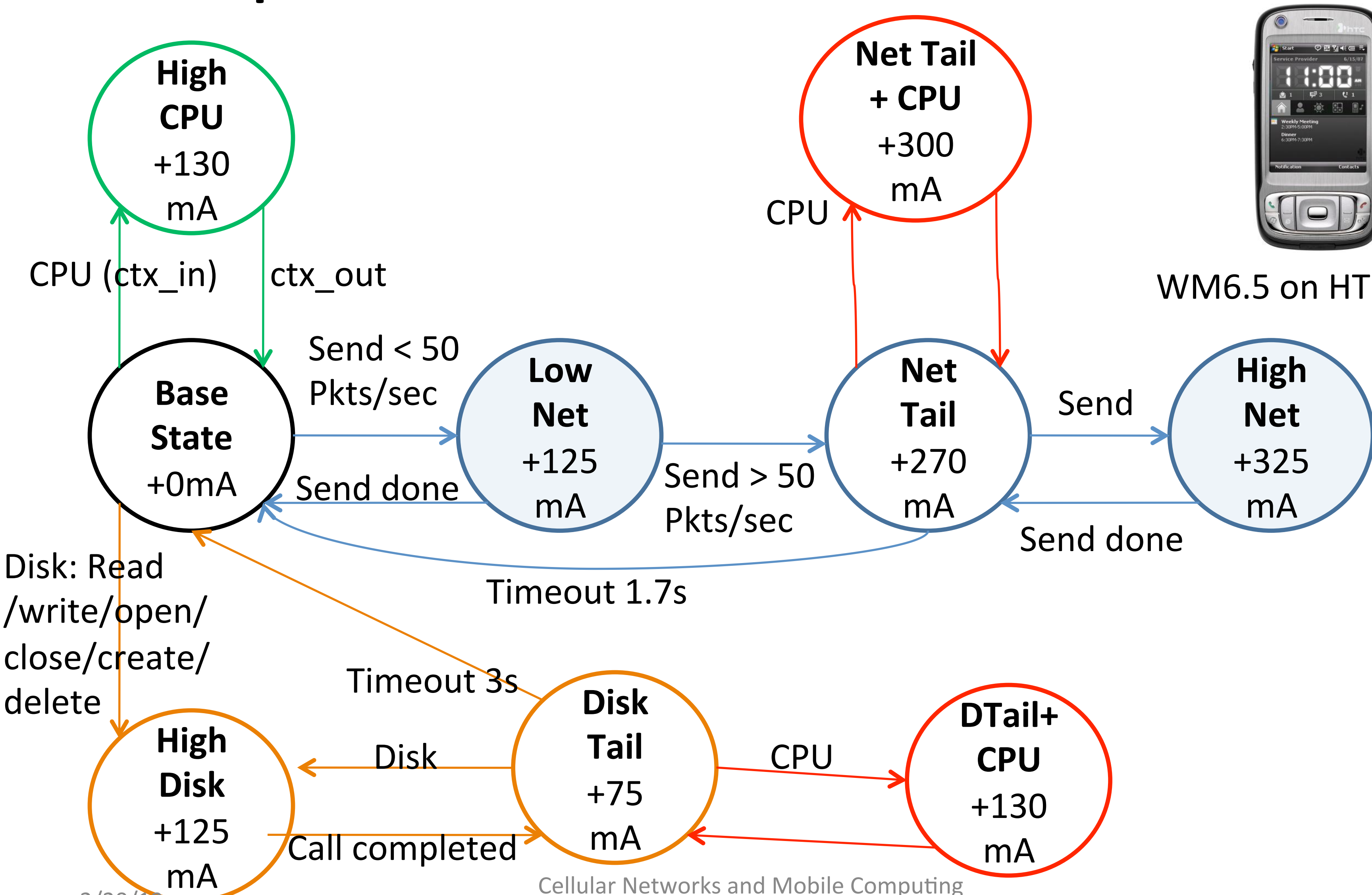
- Windows Mobile 6.5
  - Extended CeLog
- Android
  - System Tap: Logs kernel events
  - Android debugging framework: Custom logging in Dalvik VM

Evaluation: Handsets Used

HTC Tytn II	HTC Touch	HTC Magic
Win 6.5 (CE 5.2)	Win 6.5 (CE 5.2)	Android (Linux 2.6.34)

Courtesy: Pathak et al

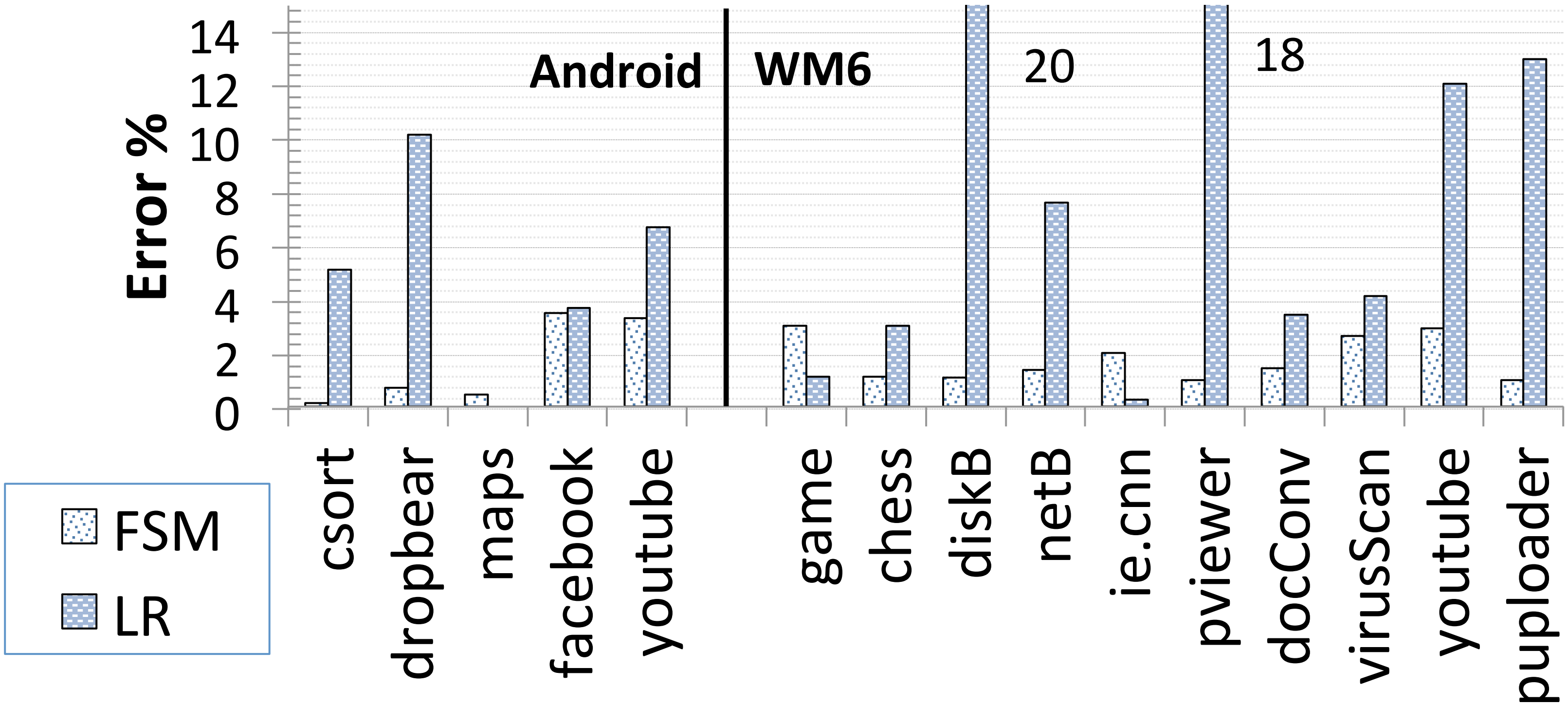
Snapshot of FSM for Entire Phone



WM6.5 on HTC Tytn II

Courtesy: Pathak et al

End-To-End Energy Estimation Error



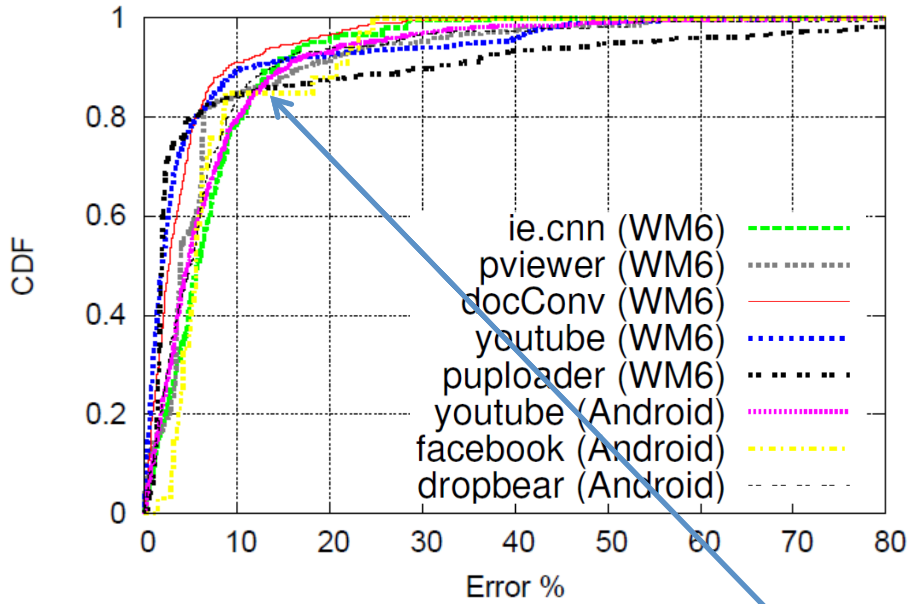
FSM: under 4%

LR: 1% – 20%

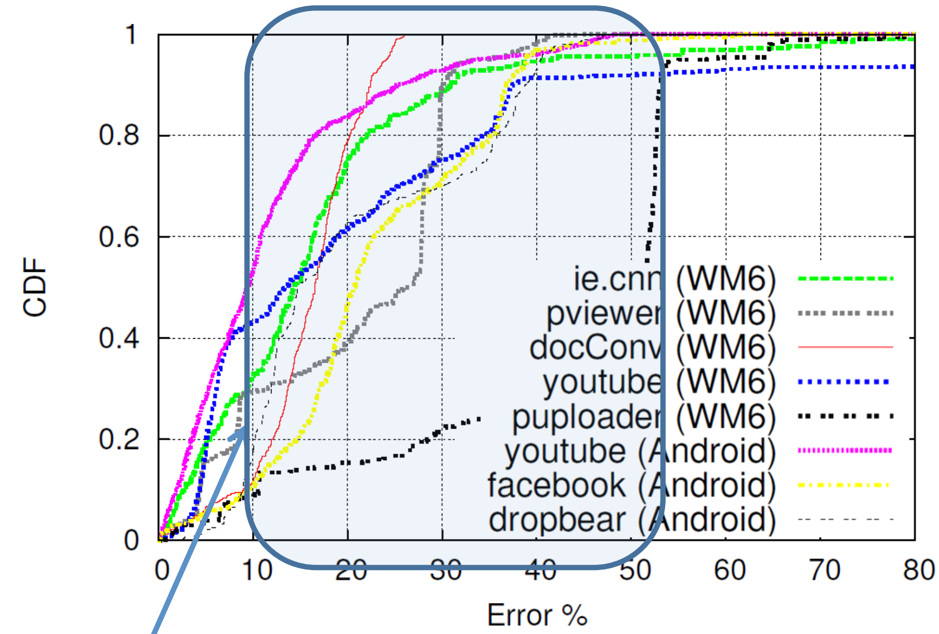
Courtesy: Pathak et al

Fine-Grained Energy Estimation

CDF of energy estimation error per 50ms time interval



FSM based on System calls



Linear Regression (State-of-art)

FSM: 80th percentile error less than 10% for all apps
LR: 10th percentile error less than 10% for all apps

Courtesy: Pathak et al

Outline

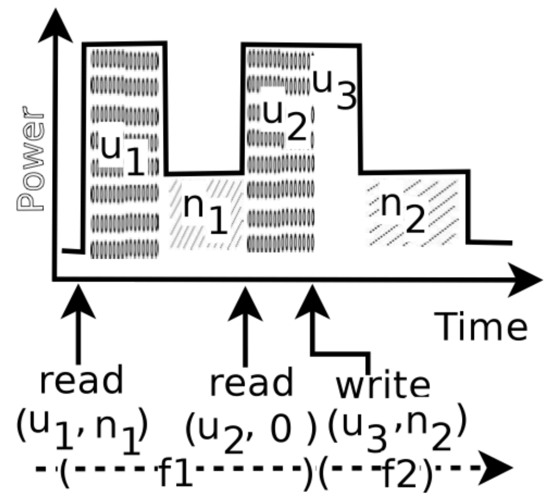
- The Rise of Ebugs
- Methods of Measuring Power Usage
- Power Models
  - Usage based
  - System call trace based
- Profiling
- Conclusion

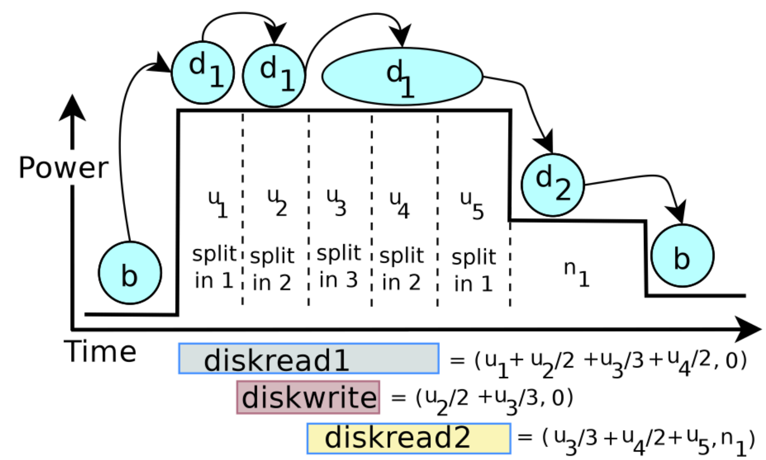
Energy Profiling

- eprof published in Eurosys 2012
- QCOM Trepn Profiler
  - Trepn leverages hardware sensors built into the Snapdragon MDP
  - Analyze power consumption of hardware blocks in the Snapdragon MDP, including:
    - CPU (system and auxiliary)
    - GPS
    - Bluetooth
    - Camera
    - Audio
    - Memory
    - Network data (optimizes data transfer frequency)

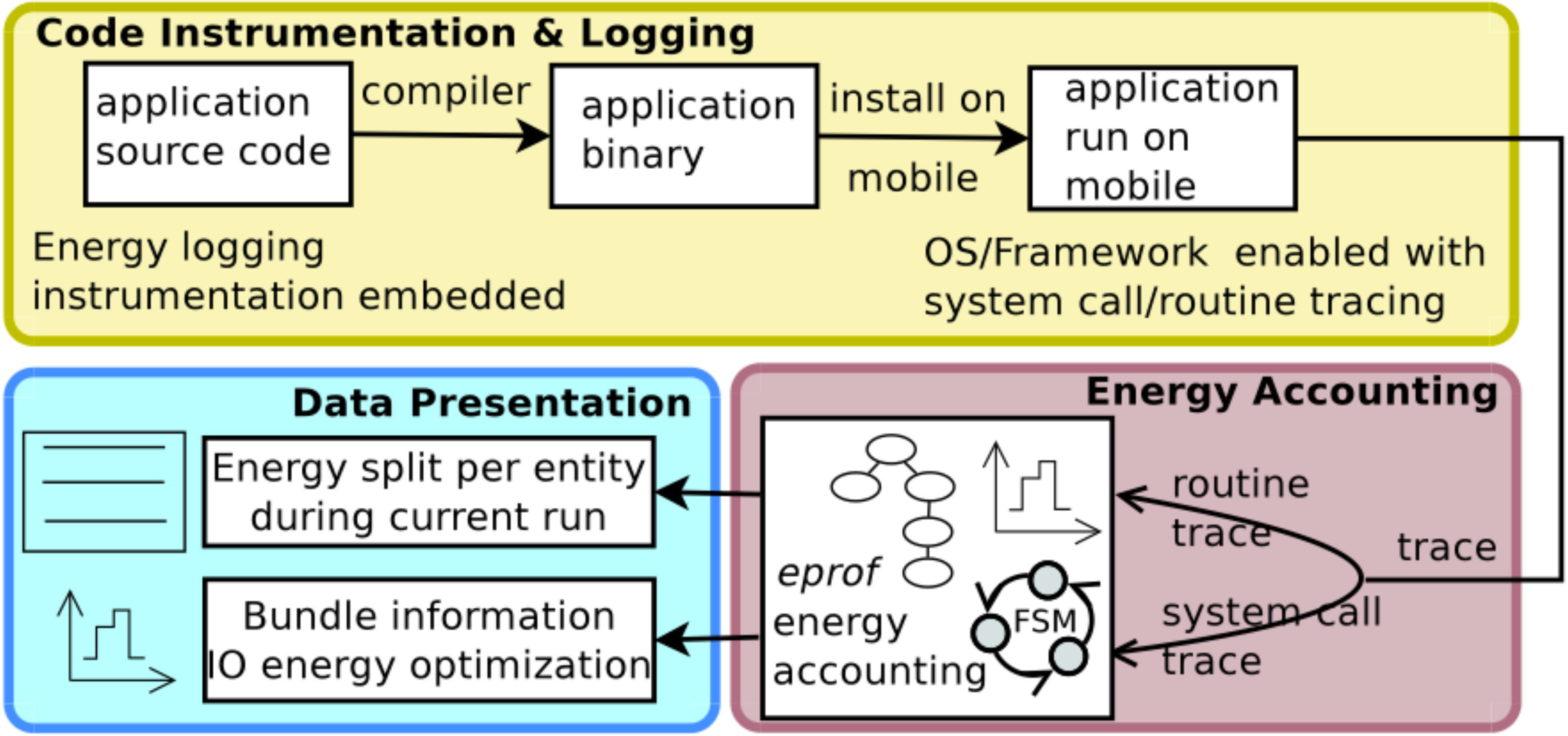
eprof

- Accounting policies for asynchronous power
  - Tail power state energy consumption: attributed to last trigger
  - Concurrent accesses: divided among multiple system calls
  - Wakelocks and exotic components: attributed to the entities that acquired the wakelock





eprof Architecture



eprof Implementation

- SDK routine tracing: extend Android routing profiling framework
  - http://developer.android.com/reference/android/os/Debug.html
- NDK routine tracing: use gprof port of NDK
  - http://code.google.com/p/android-ndk-profiler/
- System call tracing: insert ADB logging APIs in framework code and log CPU (sched.switch) scheduling event in kernel using systemtap
  - http://www.cyanogenmod.com/

eprof Evaluation

- Most energy spent on I/O

App	Run-time	#Routine calls (#Threads)	% Battery	3rd-Party Modules Used	*Where is the energy spent inside an app?*
browser	30s	1M (34)	0.35%	-	38% HTTP; 5% GUI; 16% user tracking; 25% TCP cond.
angrybirds	28s	200K (47)	0.37%	Flurry[7],Khronos[41]	20% game rendering; 45% user tracking; 28% TCP cond.
fchess	33s	742K (37)	0.60%	AdWhirl[42]	50% advertisement; 20% GUI; 20% AI; 2% screen touch
nytimes	41s	7.4M (29)	0.75%	Flurry[7],JSON[43]	65% database building; 15% user tracking; 18% TCP cond.
mapquest	29s	6M (43)	0.60%	SHW[44],AOL,JSON[43]	28% map tracking; 20% map download; 27% rendering

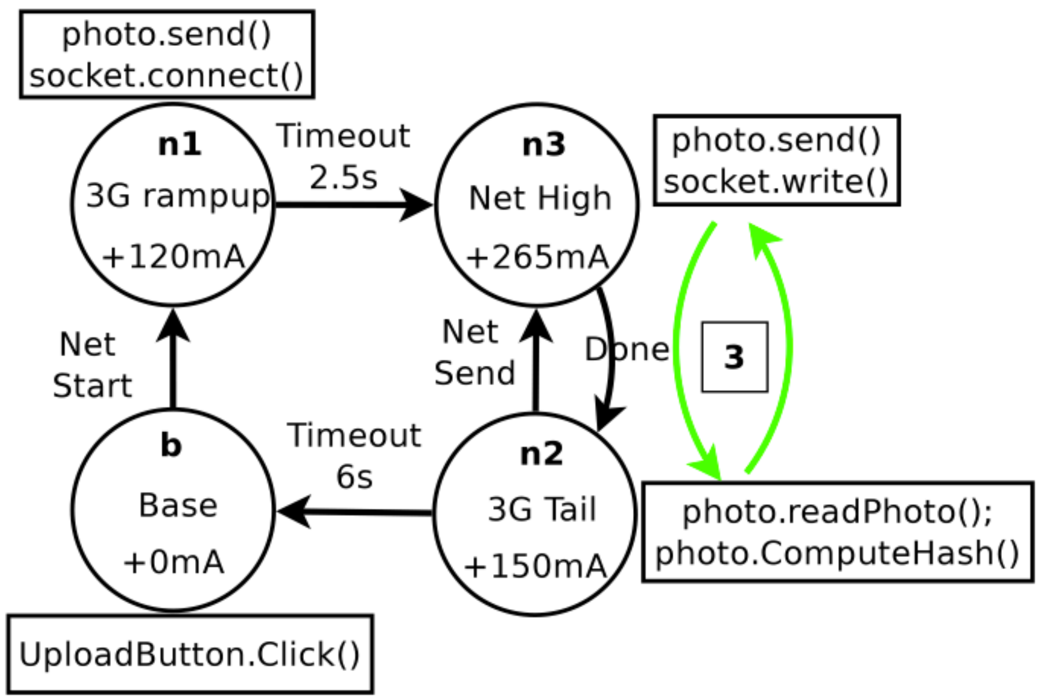
Performance Optimization

- Energy bundle: continuous period of an I/O component actively consuming power

App	Total I/O Energy	Bundles	#I/O Routines /total routines
Handset:tytn2 running WM6.5			
pslide	92%	3 (3 Disk)	2/21
pup	57%	3 (3 NET)	3/32
Handset:magic running Android			
syncdroid	50%	4 (1 NET, 3 DISK)	8/0.9K
streamer	31%	3 (3 NET)	4/1.1K
Handset:passion running Android			
browser	69%	3 (2 Net, 1 GPS)	5/3.4K
angrybirds	80%	4 (3 NET, 1 GPS)	5/2.2K
fchess	75%	2 (2 NET)	7/3.7K
nytimes	67%	2 (1 NET, 1 GPS)	16/6.8K
mapquest	72%	3 (2 NET, 1 GPS)	14/7.1K
pup	70%	1 (1 NET)	3/1.1K

Performance Optimization (Cont'd)

- Energy bundle: continuous period of an I/O component actively consuming power



Paper Contains …

- Detailed FSM construction
  - Handling special cases (CPU Frequency, WiFi Signal Strength)
  - FSM for 3 smartphones
- Detailed Accuracy Results
  - Why our model performs better than state-of-art
- Logging Overhead
  - Under 10% overhead on both the OSes
- Application: Energy Profiler
  - Call-Graph Energy profiler for smartphone apps
  - Generates source code heat map

Courtesy: Pathak et al

Conclusion and Future work

- Ebugs need to be dealt with
- Fine-grained energy modeling and profiling very important to pinpoint energy bottleneck and ebugs
  - Accounting is tricky
  - I/O energy consumption is a major part
- Display energy modeling and profiling is still lacking

Courtesy: Pathak et al

Online Resources

- ded: decompiling android application tool
  - http://siis.cse.psu.edu/ded/

Questions?