Drowsy Power Management

Matthew Lentz  James Litton  Bobby Bhattacharjee

University of Maryland
Background Energy Consumption

Mobile devices consume energy without human-interaction

Many (periodic) short-lived events executed:

- Fetch Remote Data
- Communicate with Nearby Device(s)
- Receive Push Notification
- Sample Sensor(s)
  
...
Power Trace with Events
Power Trace with Events

**Event** = Pull data from remote server over WiFi
Power Management States

- **Suspend**: System in low-power state, Memory retains contents.
- **Off**: System in low-power state.
- **On**:

![Power Consumption Graph](Image)

Event
Power Management States

- **On**: System is active.
- **Suspend**: System is in a low-power state.
- **Off**: System is powered off.

Graph:
- Power Consumption (mW) on the y-axis.
- Time on the x-axis.

Event:
- An event causes a transition from On to Suspend.
- During Suspend, memory retains its contents.
- From Suspend, the system can transition back to On.

Diagram:
- On state
- Suspend state (system in low-power state)
- Off state (powered off)

Legend:
- Power Consumption (mW)
- Time
- Event
Power Management States

- **On**: System is powered on.
- **Suspend**: System is in a low-power state. Memory retains contents.
- **Off**: System is powered off.

![Power Consumption Graph](Image)
Power Management States

- **On**: System is fully operational.
- **Suspend**: System is in a low-power state, but memory retains contents.
- **Off**: System is powered off.

**WiFi Controller Power-Save Mode**

- **Event**: Power consumption spikes when an event occurs.

**Graph**

- X-axis: Time (0-4 units).
- Y-axis: Power Consumption (mW) (0-1200 mW).

- **On** state transitions to **Suspend** state via an event.
- **Suspend** state transitions to **Off** state.
- **Off** state transitions back to **Suspend** state.
Power Trace with Events

- On
  - Event
- Suspend
- On
  - Event
Power Trace with Events

Suspend

On

Event

Power Consumption (mW)

0  200  400  600  800  1000  1200

0  2  4  6  8  10  12

Time (seconds)

Event
What Happens During a Wakeup?
What Happens During a Wakeup?

Interrupt (from RTC Alarm)
What Happens During a Wakeup?

Power Consumption (mW)

0 200 400 600 800 1000 1200

Suspend→On

Interrupt (from RTC Alarm)

Time (milliseconds)
What Happens During a Wakeup?

Suspend → On → Event

Interrupt (from RTC Alarm)
What Happens During a Wakeup?

Suspend → On → Event

Objects held by applications and drivers
When all locks released, can enter Suspend

Power Consumption (mW)

Interrupt (from RTC Alarm)

Time (milliseconds)
What Happens During a Wakeup?

Suspend $\rightarrow$ On

Event

Interrupt (from RTC Alarm)
What Happens During a Wakeup?

Power Consumption (mW)

- Suspend → On
- Event
- On → Suspend

Interrupt (from RTC Alarm)

Time (milliseconds)
What Happens During a Wakeup?

- Suspend → On
- Event
- On → Suspend

- Acquire wakelock
- Fetch update over WiFi
- Set future wakeup alarm
- Release wakelock

Interrupt (from RTC Alarm)
Transitions are Inefficient

Transitions account for 75% of total energy consumption!

[Graph showing power consumption over time with transitions labeled as Suspend→On, Event, and On→Suspend.]
Transitions are Inefficient

Transitions account for 75% of total energy consumption!

- Suspend → On: 89%
- Event: 32%
- On → Suspend: 23%

Transitions for Other Events:
- RTC Alarm (ALM): 85%
- Bluetooth Connection (BT2): 32%
- Push Notification (PSH): 89%
- Sample Accelerometer (SEN): 23%
Transitions are Inefficient

Transitions account for 75% of total energy consumption!
On $\rightarrow$ Suspend
On → Suspend

1. Flush filesystem buffers
2. Freeze all tasks
3. Suspend all devices
4. Disable non-boot CPUs
5. Set RAM to self-refresh

* Wait for interrupt
On → Suspend

1. Flush filesystem buffers
2. Freeze all tasks
3. Suspend all devices
4. Disable non-boot CPUs
5. Set RAM to self-refresh

* Wait for interrupt

Time (milliseconds)
Suspend → On
Suspend → On

1. Enable CPUs
2. Resume all devices
3. Thaw all tasks
Suspend → On

1. Enable CPUs
2. Resume all devices
3. Thaw all tasks
Drowsy Power Management

Wake up only what is necessary
Drowsy Power Management

Wake up only what is necessary

Necessary for Pull Data:

{ }
Drowsy Power Management

Wake up only what is necessary

Necessary for Pull Data:

- Application
- Alarm Device
- System Services
- WiFi Device
Drowsy Power Management

Wake up only what is necessary

Necessary for Pull Data:
- Application
- Alarm Device
- WiFi Device

Unnecessary:
- USB Device
- Calendar App
- Battery Monitoring Device
- Bluetooth Device
- Power Regulator Devices
- Input Devices
- Power Regulators
- SD Card Device
- ...
Drowsy Power Management

Construct minimal “wake set” of tasks and devices
Drowsy Power Management

Construct minimal "wake set" of tasks and devices

- Smallest set that maintains correct behavior
Drowsy Power Management

Construct **minimal “wake set”** of tasks and devices

- Smallest set that **maintains correct behavior**
- Expand **on-demand** as event progresses
Drowsy Power Management

Construct **minimal “wake set”** of tasks and devices

- Smallest set that maintains correct behavior
- Expand on-demand as event progresses

**Constraint:** No modifications to user-space
Transitions To/From Drowsy

Drowsy

Thaw all tasks 3
Resume all devices 2
Enable CPUs 1

Suspend

1. Flush filesystem buffers
2. Freeze all tasks
3. Suspend all devices
4. Disable non-boot CPUs
5. Set RAM to self-refresh
Transitions To/From Drowsy

Thaw previously running tasks
Resume all devices
Enable CPUs

Flush filesystem buffers
Freeze all tasks
Suspend all devices
Disable non-boot CPUs
Set RAM to self-refresh
Transitions To/From Drowsy

On CPU or Run Queue

Thaw previously running tasks
Resume all devices
Enable CPUs

1. Flush filesystem buffers
2. Freeze all tasks
3. Suspend all devices
4. Disable non-boot CPUs
5. Set RAM to self-refresh
Transitions To/From Drowsy

Wake Set = \{ <Prev. Running Tasks> \}

Thaw previously running tasks
Resume all devices
Enable CPUs

Drowsy
1. Flush filesystem buffers
2. Freeze all tasks
3. Suspend all devices
4. Disable non-boot CPUs
5. Set RAM to self-refresh

Suspend
Transitions To/From Drowsy

Wake Set =
\{ <Prev. Running Tasks> \}

- Thaw previously running tasks
- Resume all devices
- Enable CPUs

1. Flush filesystem buffers
2. Freeze all tasks
3. Suspend all devices
4. Disable non-boot CPUs
5. Set RAM to self-refresh
Transitions To/From Drowsy

Wake Set =
{ <Prev. Running Tasks>, Task, Device, ... }

- Thaw previously running tasks 3
- Resume all devices 2
- Enable CPUs 1
- 1. Flush filesystem buffers
- 2. Freeze all tasks
- 3. Suspend all devices
- 4. Disable non-boot CPUs
- 5. Set RAM to self-refresh
Transitions To/From Drowsy

Wake Set = 
{ <Prev. Running Tasks>, Task, Device, ... }

Thaw previously running tasks 3
Resume all devices 2
Enable CPUs 1

Flush filesystem buffers 1
Freeze tasks in wake set 2
Suspend devices in wake set 3
Disable non-boot CPUs 4
Set RAM to self-refresh 5
Constructing the Wake Set

Task States:  
- Run (CPU)  
- Run (I/O)  
- Run (Device I/O)

Time

Task A

- open() named pipe
- read() from pipe
- ioctl() command to device
Constructing the Wake Set

Task States: Run (CPU) Run (I/O) Run (Device I/O) Blocked

Task A
- open() named pipe
- read() from pipe
- Blocks waiting on condition to be met
Constructing the Wake Set

Task States: Run (CPU) Run (I/O) Run (Device I/O) Blocked

Task B

write() to pipe

Task A

open() named pipe

read() from pipe

Blocks waiting on condition to be met

ioctl() command to device

Satisfies condition
Constructing the Wake Set

Task States:  
- Run (CPU)  
- Run (I/O)  
- Run (Device I/O)  
- Blocked

Time

Task B  

Task A  
- open() named pipe
- read() from pipe
- Blocks waiting on condition to be met

write() to pipe

open() named pipe

read() from pipe

Blocks waiting on condition to be met
Constructing the Wake Set

Task States:  
- Run (CPU)  
- Run (I/O)  
- Run (Device I/O)  
- Blocked  
- Frozen

Time

Task B

Task A

Suspended
Constructing the Wake Set

Task States:  
- Run (CPU)  
- Run (I/O)  
- Run (Device I/O)  
- Blocked  
- Frozen

Time

Task B  Task A

Transition to Suspend (All wakelocks released)
Constructing the Wake Set

Task States:
- Run (CPU)
- Run (I/O)
- Run (Device I/O)
- Blocked
- Frozen

Time

Task B

Task A

Transition to **Suspend**
(All wakelocks released)

Transition to **Drowsy**
Constructing the Wake Set

Task States:
- Run (CPU)
- Run (I/O)
- Run (Device I/O)
- Blocked
- Frozen

Wake Set =

Time

Task B

Task A

Suspended
Constructing the Wake Set

Task States:  
- Run (CPU)
- Run (I/O)
- Run (Device I/O)
- Blocked
- Frozen

Wake Set = { Task B }

Time

Task B

Task A

Suspended
Constructing the Wake Set

Task States:  
- Run (CPU)
- Run (I/O)
- Run (Device I/O)
- Blocked
- Frozen

Wake Set = 

- { Task B }
- { Task B, Task A }
Constructing the Wake Set

Task States: Run (CPU)  Run (I/O)  Run (Device I/O)  Blocked  Frozen

Time

Task B

Task A

Wake Set =

{ Task B }  
{ Task B, Task A }  
{ Task B, Task A, Device }
Drowsy Wake Set: Pull Data
Drowsy Wake Set: Pull Data

- **Task**: ActivityManager, Binder_1, Binder_2, Binder_3, Binder_4, Binder_1, Binder_1, Worker, Compiler, log_main, wcns_wlan.0, wlan0
- **Device**: cpu1, cpu2, cpu3, Binder_1
- **IRQ**: IRQ 48, IRQ 362, IRQ 479, platform, rtc-pm8xxx, /dev/alarm

Time (ms):
- CPU: 0 ms, 4 ms, 6 ms, 26 ms, 41 ms, 91 ms, 98 ms, 13 ms
- Device:
  - Suspend (Sync Filesystem Buffers)
  - Resume (Thaw Running Tasks)

Diagram shows the flow of tasks, devices, and IRQs with timelines indicating when tasks are executed or suspended.
Drowsy Wake Set: Pull Data

15 tasks thawed & 16 devices resumed

1.8% 1.9%
Implementation

Implemented Drowsy within Android kernel

Platform: Google Nexus 4 ("Mako")
Version: 4.2.2 (Fork of Linux 3.4)
SLOC: ~4,600
Instrumenting Android

For determining when to add tasks/devices to wake set
Instrumenting Android

For determining when to add tasks/devices to wake set

**Diagram:**
- **FS Calls (file_operations)**
- Sockets (net_device_ops)
- Attributes (device_attributes)
- MMIO (mmap)

**Instrumented by Drowsy**

**Blocks:**
- Tasks
- Drivers
- Hardware

**Connections:**
- IPC (ashmem, signals)
- Device Classes
- IRQs
- Bus / MMIO
- Wait Queues (try_to_wake_up)
Example: `file_operations`

```
struct file_operations d_fops:
  .open = &drv_open
  .read = &drv_read
  .write = &drv_write
```
Example: file_operations

```c
struct file_operations d_fops:
  .open = &drv_open
  .read = &drv_read
  .write = &drowsy_write
  .write_impl = &drv_write
```
Example: file_operations

```c
struct file_operations d_fops:
  .open = &drv_open
  .read = &drv_read
  .write = &drowsy_write
  .write_impl = &drv_write

int drowsy_write(File *f, ...)
  Device *d = fileToDevice(f)
  if(d->state == Suspended)
      resumeDevice(d)
  f->f_ops.write_impl(f, ...)
```
Evaluation: Benchmarking

- On → Suspend
- Drowsy → Suspend
- Release Wakelock
- Acquire Wakelock
- Handle I/O Event

ALM Set a future wakeup alarm
BT2 Accept incoming Bluetooth connection
PUL Fetch weather update (and set alarm)
PSH Receive incoming push notification
SEN Sample the accelerometer (and set alarm)
Evaluation: Measurement

\[ I_{\text{Phone}} = \frac{V_{\text{Shunt}}}{R_{\text{Shunt}}} \]
Evaluation: Measurement

\[ I_{\text{Phone}} = \frac{V_{\text{Shunt}}}{R_{\text{Shunt}}} \]
Evaluation: Measurement

\[ I_{\text{Phone}} = \frac{V_{\text{Shunt}}}{R_{\text{Shunt}}} \]
Evaluation: Measurement

\[ I_{\text{Phone}} = \frac{V_{\text{Shunt}}}{R_{\text{Shunt}}} \]
Improvement: Wakeup Events

The diagram compares the performance of different devices and configurations in terms of time speedup and energy efficiency. The devices are categorized into Android, Android Plus, Android Plus (1 Core, Powersave), Drowsy, and Drowsy (1 Core, Powersave). The categories are ALM, BT2, PUL, PSH, and SEN. The x-axis represents time speedup and energy efficiency, ranging from 6x to 1x.
Improvement: Wakeup Events

Drowsy is 1.5-5x as energy efficient

Drowsy is 1.5-5x as energy efficient
Improvement: Wakeup Events

cpuidle entered deep idle states more often in the Drowsy state

Drowsy is 1.5-5x as energy efficient
Improvement: Wakeup Events

- Android
- Android Plus
- Android Plus (1 Core, Powersave)
- Drowsy
- Drowsy (1 Core, Powersave)

![Graph showing Time Speedup and Energy Efficiency for different Android versions and power modes.](image-url)
Wakeup Cycle: Pull Data
Wakeup Cycle: Pull Data

**Android**

- **Power Consumption (mW)**
  - Y-Axis: 0 to 1200
  - X-Axis: 0 to 200 milliseconds

- **Events**
  - Suspend → On
  - On → Suspend

**Drowsy**

- **Power Consumption (mW)**
  - Y-Axis: 0 to 1200
  - X-Axis: 0 to 200 milliseconds

- **Events**
  - Event & Wake Set
  - Drowsy
  - Suspend

Graphs showing power consumption over time for Android and Drowsy modes, with different states and events indicated.
Wakeup Cycle: Pull Data

Android

Drowsy

WiFi Tail Energy
Improvement: Battery Life
Improvement: Battery Life

Converges to 0% as the interval increases
Improvement: Battery Life

Benefits of Drowsy aggregate across all applications on device

Converges to 0% as the interval increases
Summary

Existing power management not optimized for short-lived events

Drowsy wakes up the minimal set of tasks and devices is 1.5 - 5x as energy efficient for short-lived events

Source code is available at:

www.cs.umd.edu/projects/drowsy/