Drowsy Power Management

Matthew Lentz  James Litton  Bobby Bhattacharjee
University of Maryland
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

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

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

Event

**Graph:**
- Power Consumption (mW) vs. Time
- `0` to `1200` mW range
- `0` to `4` Time units

Flowchart:
- **On** to **Suspend**
- **Suspend** to **On**
- **Suspend**: System in low-power state
- **Suspend**: Memory retains contents
Power Management States

- **On**: System is active and consumes power.
- **Suspend**: System is in a low-power state, consuming minimal power.
- **Off**: System is powered down, consuming no power.

- Memory retains contents during suspend state.

Graph shows power consumption over time with an event occurring at specific points.
Power Management States

On

Suspend

Off

System in low-power state

Memory retains contents

Event

Power Consumption (mW)

Time
Power Management States

- On
- Suspend
- Off

Event

System in low-power state
Memory retains contents

WiFi Controller
Power-Save Mode
Power Trace with Events

Suspend

On

Event

On

Event

Power Consumption (mW)

Time (seconds)
Power Trace with Events

- Suspend
- On
- Event

Power Consumption (mW)

Time (seconds)
What Happens During a Wakeup?
What Happens During a Wakeup?

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

Interrupt (from RTC Alarm)

Power Consumption (mW)

Time (milliseconds)
What Happens During a Wakeup?

Suspend → On

Event

Interrupt (from RTC Alarm)

Power Consumption (mW)

Time (milliseconds)
What Happens During a Wakeup?

- Suspend $\rightarrow$ On
- Event

Objects held by applications and drivers

When all locks released, can enter **Suspend**
What Happens During a Wakeup?

Suspend $\rightarrow$ On Event

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

Suspend → On → Event

Interrupt (from RTC Alarm)

On → Suspend

Power Consumption (mW) vs. Time (milliseconds)
What Happens During a Wakeup?

During a wakeup, the following events occur:

1. **Acquire wakelock**
2. **Fetch update over WiFi**
3. **Set future wakeup alarm**
4. **Release wakelock**

The diagram illustrates the power consumption over time, with different phases labeled as **Suspend → On**, **Event**, and **On → Suspend**. The power consumption is measured in milliwatts (mW). The diagram also highlights the interrupt from the RTC Alarm at the beginning.
Transitions are Inefficient

Transitions account for 75% of total energy consumption!
Transitions are Inefficient

Transitions account for 75% of total energy consumption!

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!

![Diagram showing power consumption over time with transitions labeled as Suspend→On, Event, and On→Suspend.](image-url)
On $\rightarrow$ Suspend

![Graph showing power consumption vs. time, with 'On' and 'Suspend' states indicated by different colors.](image)
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

[Graph showing time (milliseconds) with numbered steps 1 to 5 indicated]
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
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
- Bluetooth Device
- Battery Monitoring Device
- Power Regulator Devices
- Power Regulators
- Input Devices
- 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 a **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

- Thaw all tasks
- Resume all devices
- Enable CPUs

Drowsy

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

Drowsy

- Resume all devices
- Enable CPUs

Suspense

- Thaw previously running tasks
- Resume all devices

Steps:
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

On CPU or Run Queue

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

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

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

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

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

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

Drowsy

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

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

Task A

- `open()` named pipe
- `read()` from pipe
- `ioctl()` command to device

Time
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

Time

Task B
- write() to pipe
- Satisfies condition

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

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

Task B

Task A

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

Time
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 }
Constructing the Wake Set

Wake Set =

{ Task B, Task A }

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

Task B

Task A

Time

Suspended
Constructing the Wake Set

WAKE SET =

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

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

Instrumented by Drowsy

Tasks

Drivers

Devices

Hardware

IRQs

generic_handle_irq

Bus / MMIO

Wait Queues *(try_to_wake_up)*

IPC (ashmem, signals)
Example: file_operations

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

\[
\text{struct file_operations d_fops:}
\]

- .open = &drv_open
- .read = &drv_read
- .write = &drowsy_write
- .write_impl = &drv_write
Example: file_operations

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

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

LED Toggled On
(Software Timestamp)

Battery

Power Supply

1M\(\Omega\)

Phone

Function Generator

DMM

\( V_{\text{Photo}} \)

\( V_{\text{Phone}} \)

\( V_{\text{Shunt}} \)

\( I_{\text{Phone}} \)

Time
Improvement: Wakeup Events

Time Speedup

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

Energy Efficiency

ALM
BT2
PUL
PSH
SEN
Improvement: Wakeup Events

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

![Graph showing time speedup and energy efficiency improvements for different applications (ALM, BT2, PUL, PSH, SEN) across various operating modes: Android, Drowsy, Android Plus, Androic Plus (1 Core, Powersave).](image-url)
Wakeup Cycle: Pull Data

Android

Power Consumption (mW)

Suspend → On

On → Suspend

Event

Time (milliseconds)
Wake up Cycle: Pull Data

Android

Power Consumption (mW)

Time (milliseconds)

Suspend → On

On → Suspend

Drowsy

Power Consumption (mW)

Time (milliseconds)

Suspend → Drowsy

Drowsy → Suspend
**Wake Up Cycle: Pull Data**

**Android**

- **Suspend → On**
- **On → Suspend**

**Drowsy**

- **Event & Wake Set**
- **Suspend**
- **WiFi Tail Energy**
Improvement: Battery Life

![Graph showing the improvement in battery life over time for different scenarios. The x-axis represents the interval between events on a logarithmic scale, while the y-axis represents the percentage improvement in battery life. The graph includes lines for SEN, PSH, ALM, PUL, and BT2, each with a different color and style. The lines show the decrease in battery life improvement over time.]
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: