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

Instructor: Li Erran Li
(lel2139@columbia.edu)

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

4/9/2012:OS Support for Energy and Sensor Management (or Rethinking Mobile OS)

Rethinking Mobile OS

- What abstraction should mobile OS provide to apps?
  - Should the OS provide fine-grained battery power management?
  - Should the OS provide high level contextual information inferred from low level sensor information?
  - Should social network apps be in the OS? An OS system call to send a tweet is much more efficient.

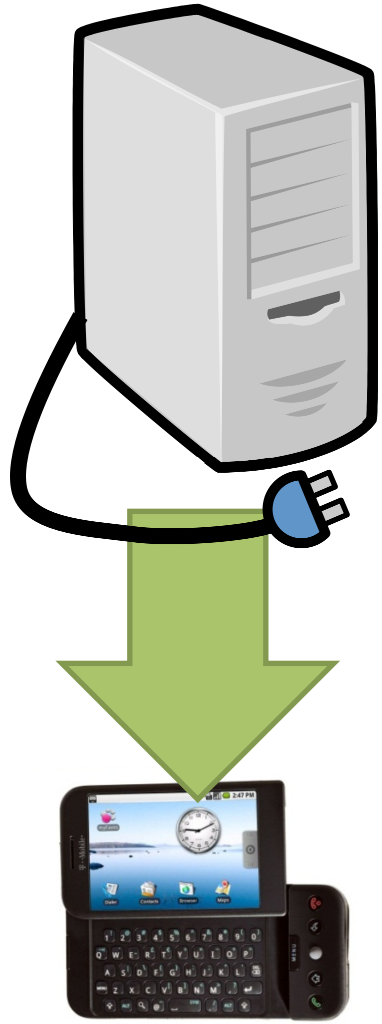
Energy Management in Mobile Devices with the Cinder Operating System

Arjun Roy, Stephen M. Rumble, *Ryan Stutsman*,
Phil Levis, and David Mazières
Stanford University

Nickolai Zeldovich
MIT CSAIL

The State of Smartphones

- Smartphones are a dominant computing platform
- Energy is the limiting resource on smartphones
- OSes don't provide any **control** over it



Cinder: Rethinking the Mobile OS

- Track application energy consumption

- **Fine-grained** energy control primitives
  - For users
    - Limit application energy consumption
  - **For applications**
    - Use app specific knowledge to manage energy
    - Even within process boundary

Real-world Implementation

- Runs on the HTC Dream (T-Mobile G1)
- Based on HiStar
- ~15,000 lines of code excluding drivers
- Working
  - Display, keyboard
  - Texting
  - 3G Data Radio
  - Answering/Placing Calls (no audio)
- Also runs on x86_64 desktops/laptops

Existing Approaches

- Android
  - Provides visibility
    - Estimates energy consumption for apps
  - **No control**

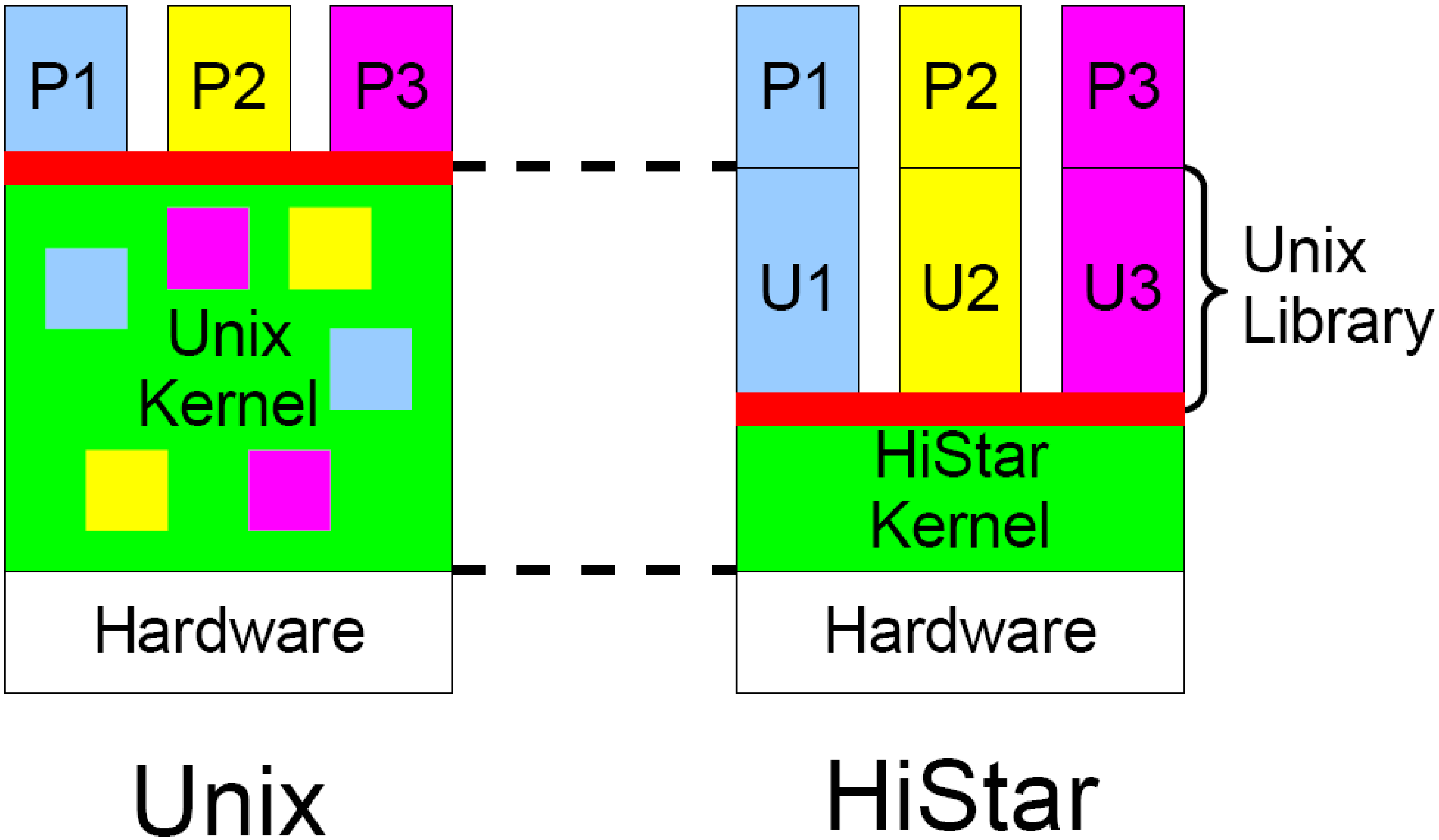
- Prior research (ECOSystem)
  - Similar visibility
  - Simple control
    - Try to meet target battery lifetime on a laptop

Outline

- **New Abstractions for Control**
- Examples
- The Problem of Closed Platforms
- Cinder-Linux

HiStar Overview

- Make all state explicit, track all communication

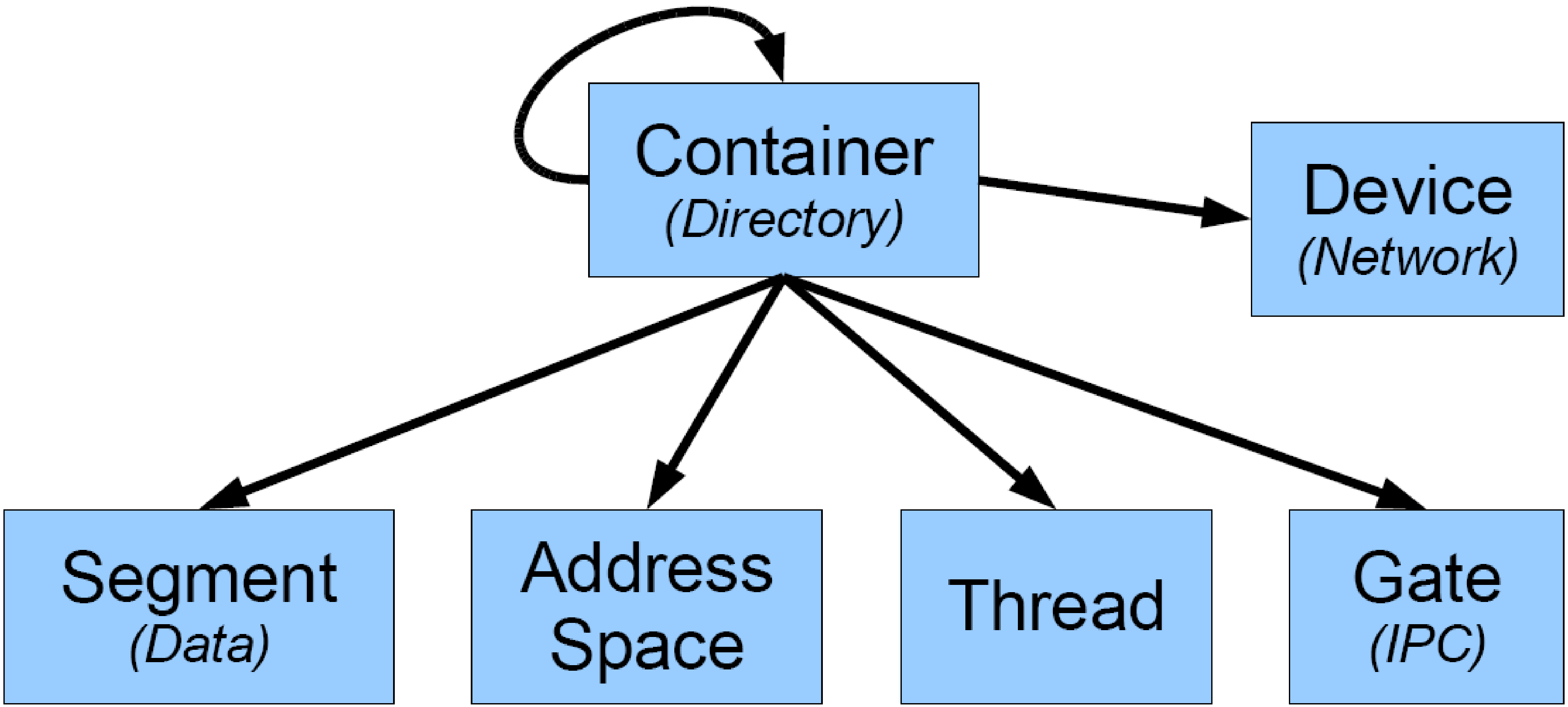


HiStar Overview (Cont'd)

- Narrow kernel interface, few comm. channels
  - Minimal mechanism: enough for a Unix library
  - Strong control over information flow

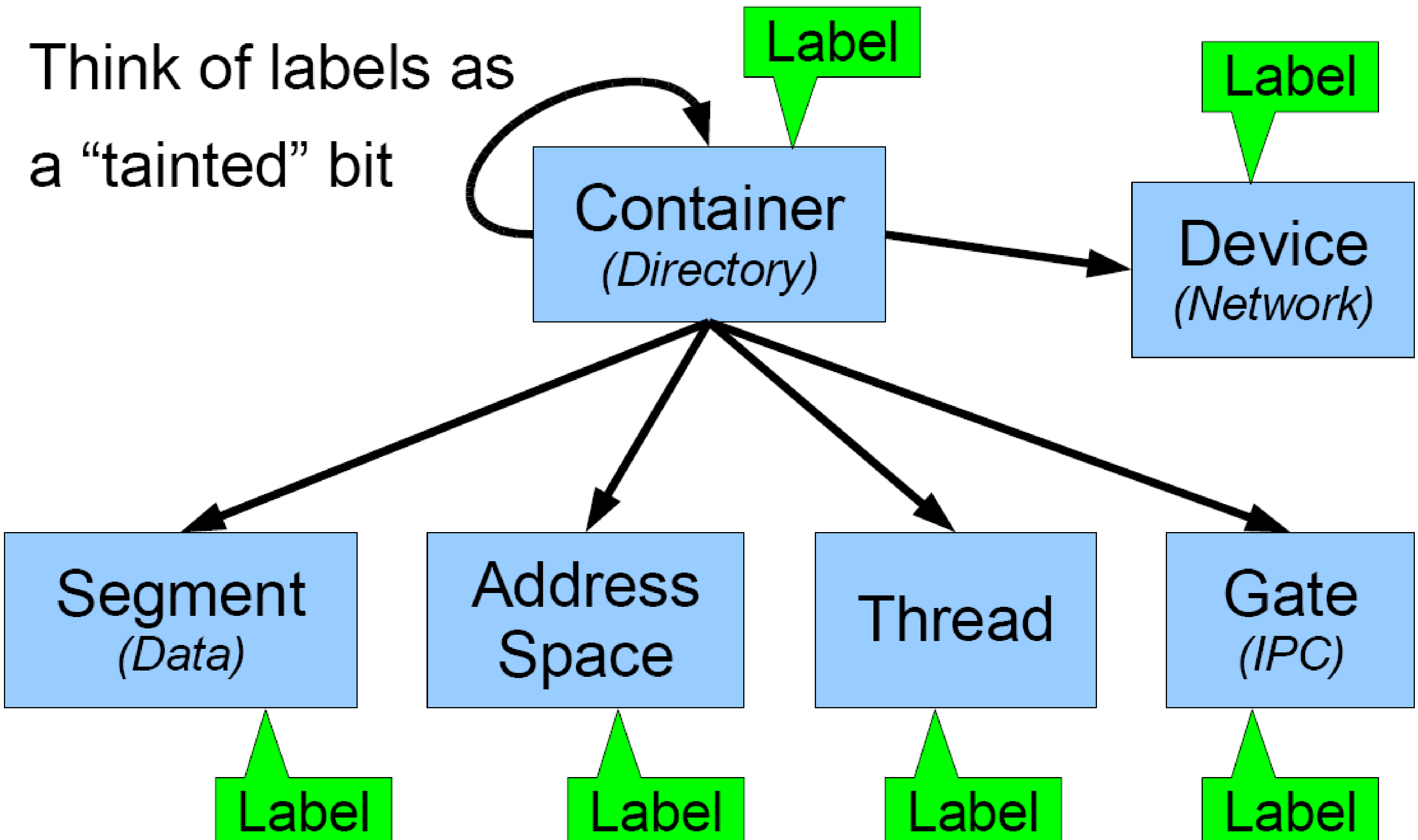
- Unix support implemented as user-level library
  - Unix communication channels are made explicit, in terms of HiStar's mechanisms
  - Provides control over the gamut of Unix channels

HiStar kernel objects



HiStar kernel objects

Think of labels as a "tainted" bit



Power modeling

- Active research area
- Measure offline in isolation
  - Device states
  - System calls
- Bill threads online using offline analysis
  - CPU
  - Data radio
  - GPS
  - Etc.

Energy Control Policies

- **Reserves** limit **quantity** of energy use
  – Subdivision and isolation between apps

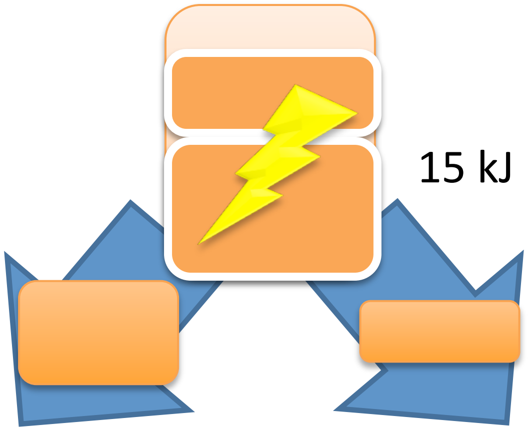
- **Taps** limit **rate** of energy use
  – Enforces “lifetime” type policies

Reserves

15 kJ

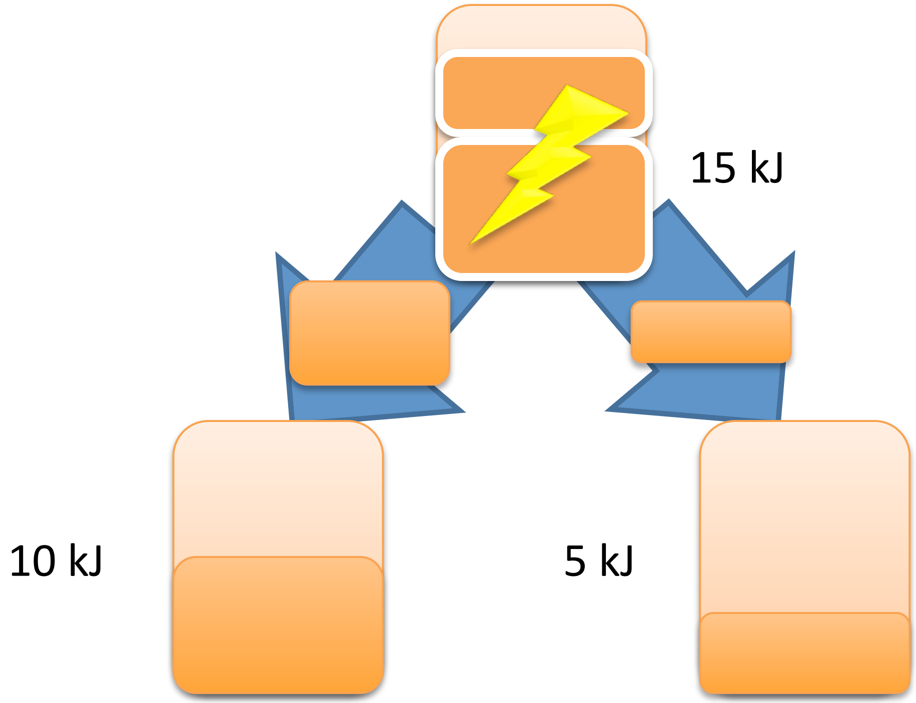
Reserves

- Virtualized batteries
  - Subdivide energy



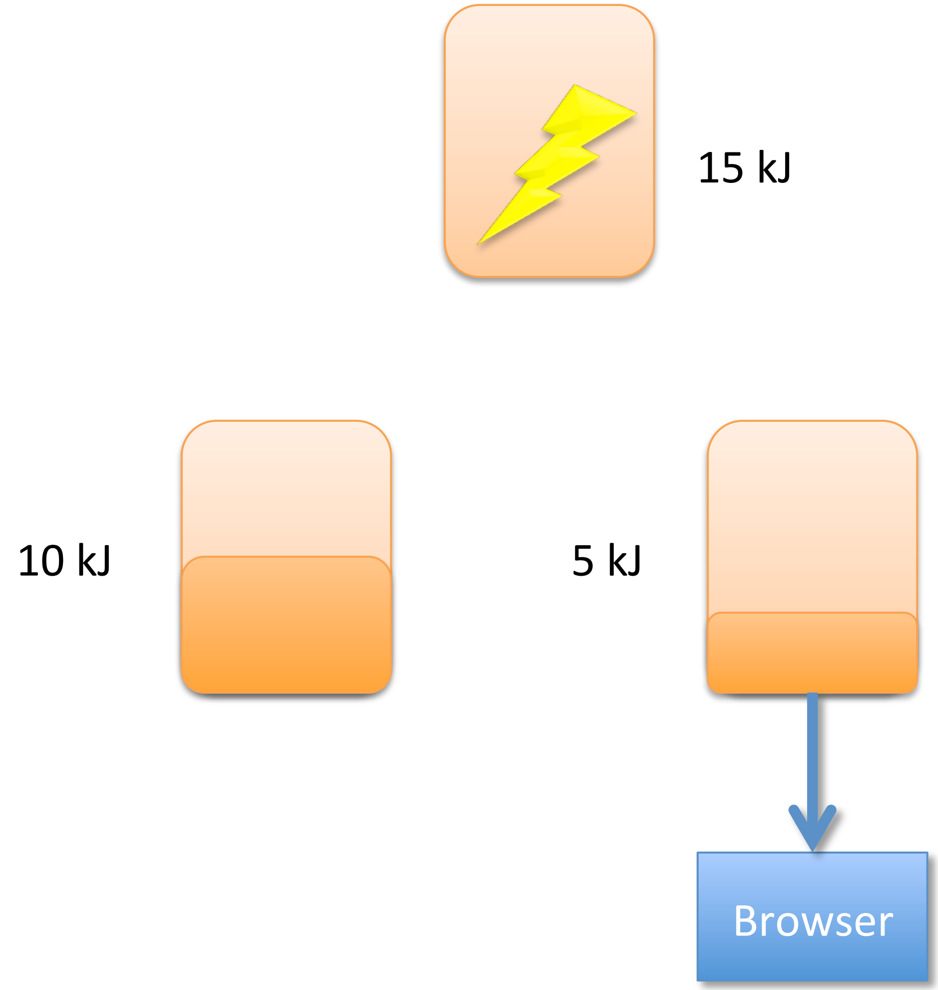
Reserves

- Virtualized batteries
  - Subdivide energy



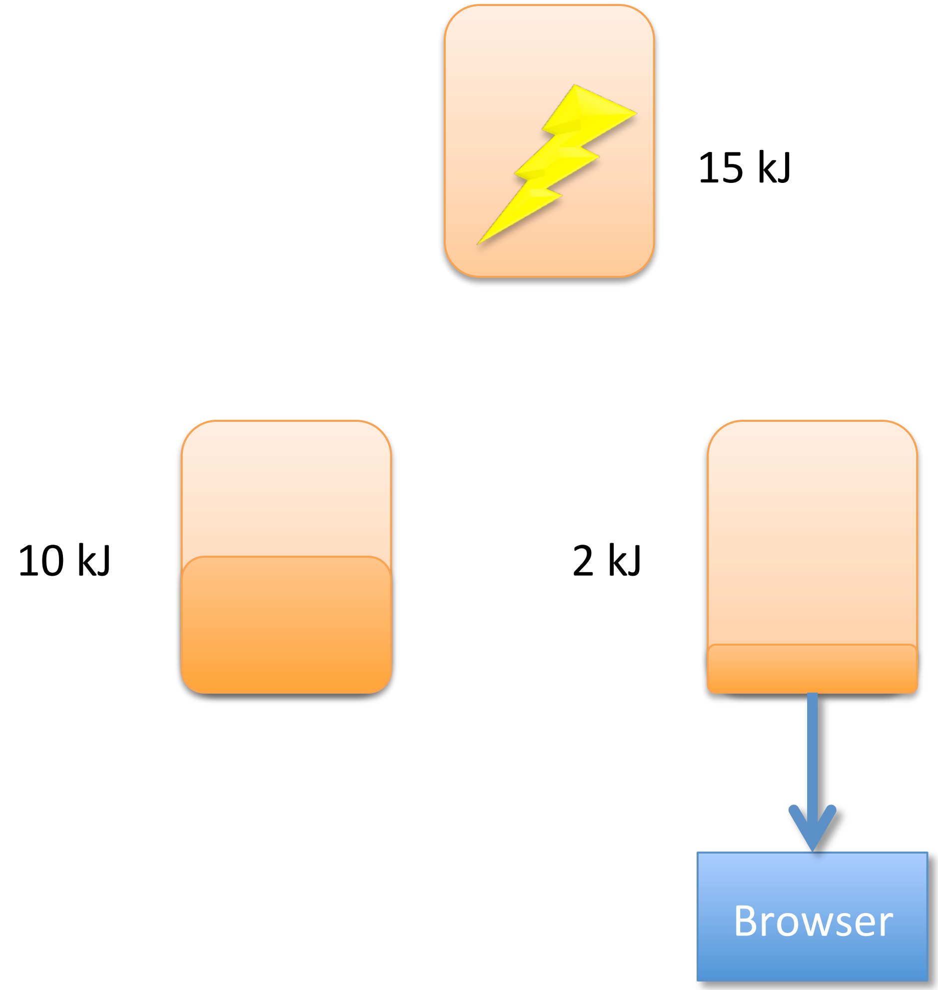
Reserves

- All threads run in the context of a reserve
  - Accounting
  - Control



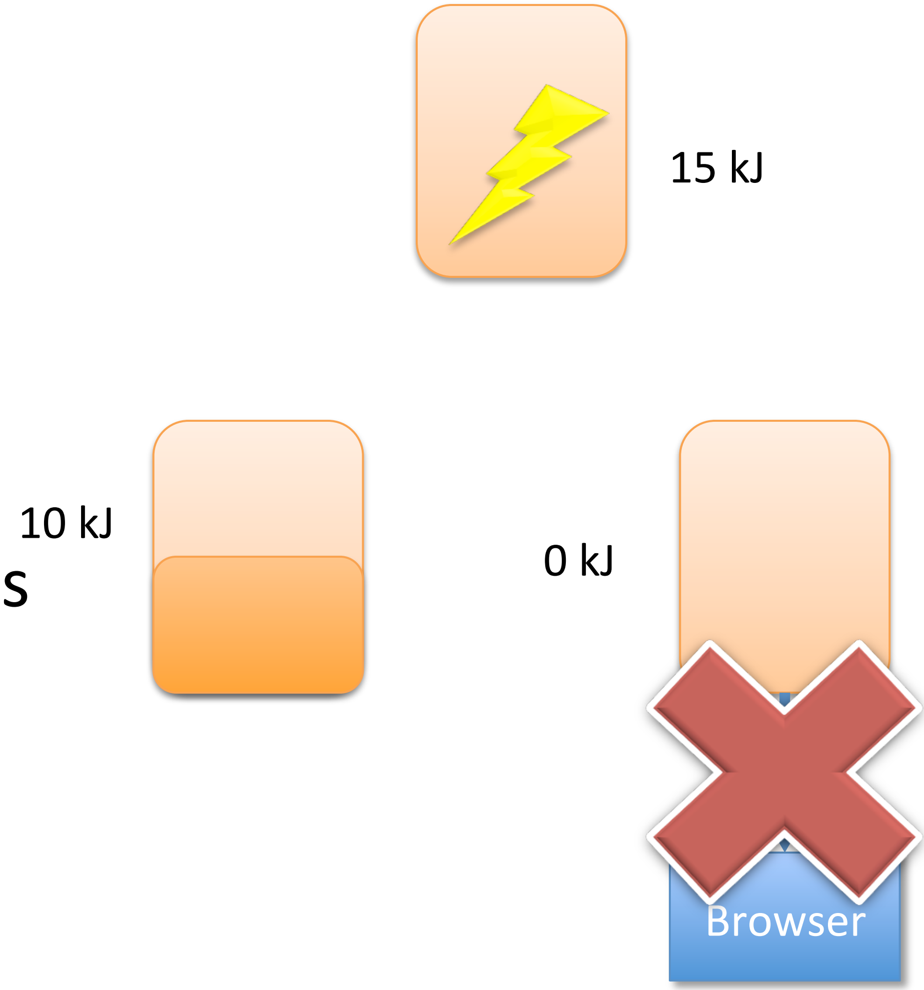
Reserves

- All threads run in the context of a reserve
  - Accounting
  - Control



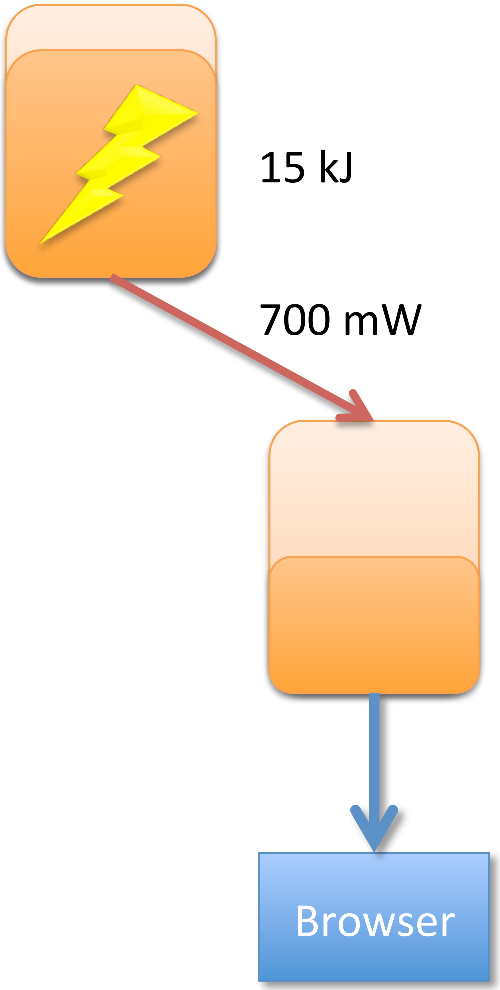
Reserves

- All threads run in the context of a reserve
  - Accounting
  - Control
- De-schedule threads with exhausted reserves



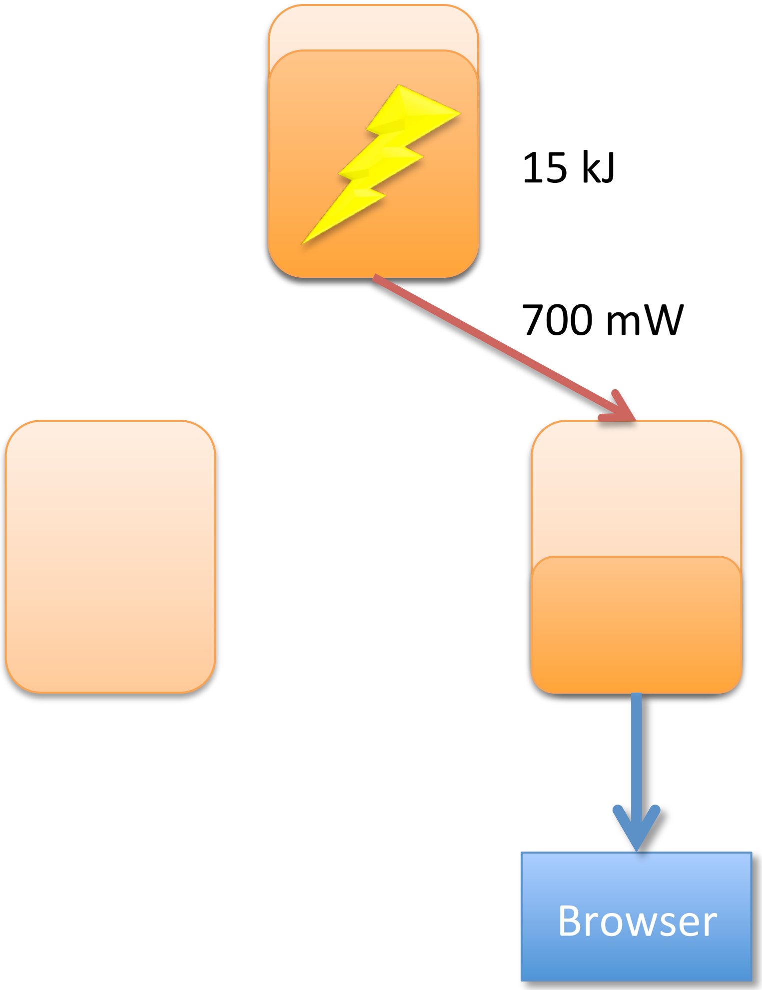
Taps

- A rate transfer between reserves



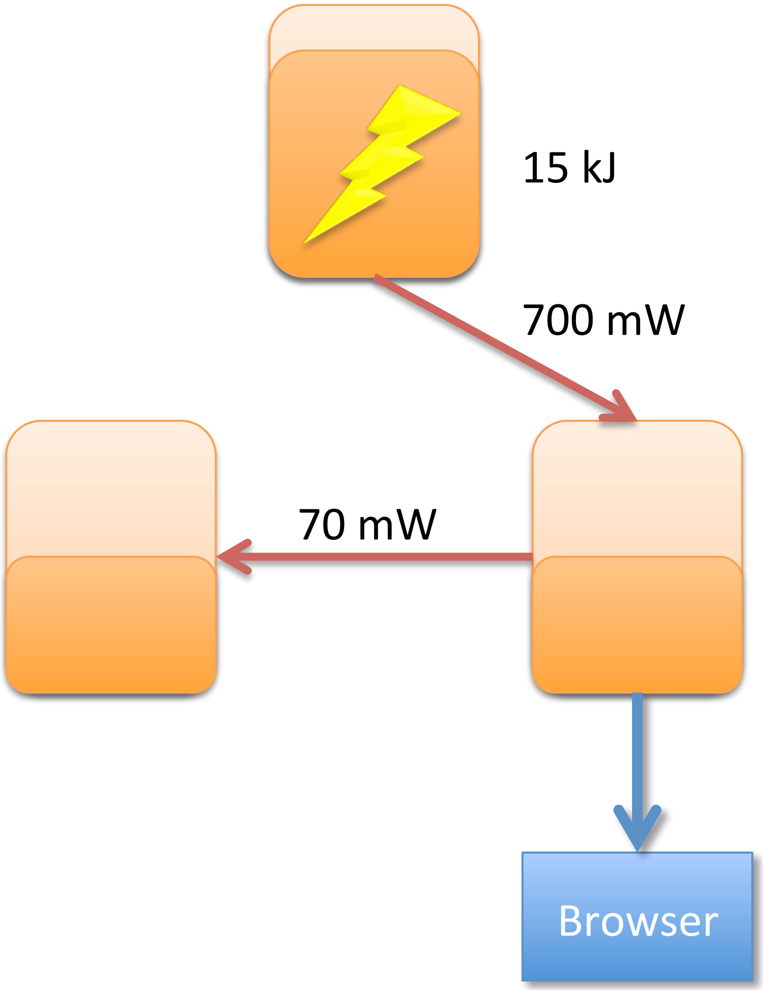
Taps

- A rate transfer between reserves



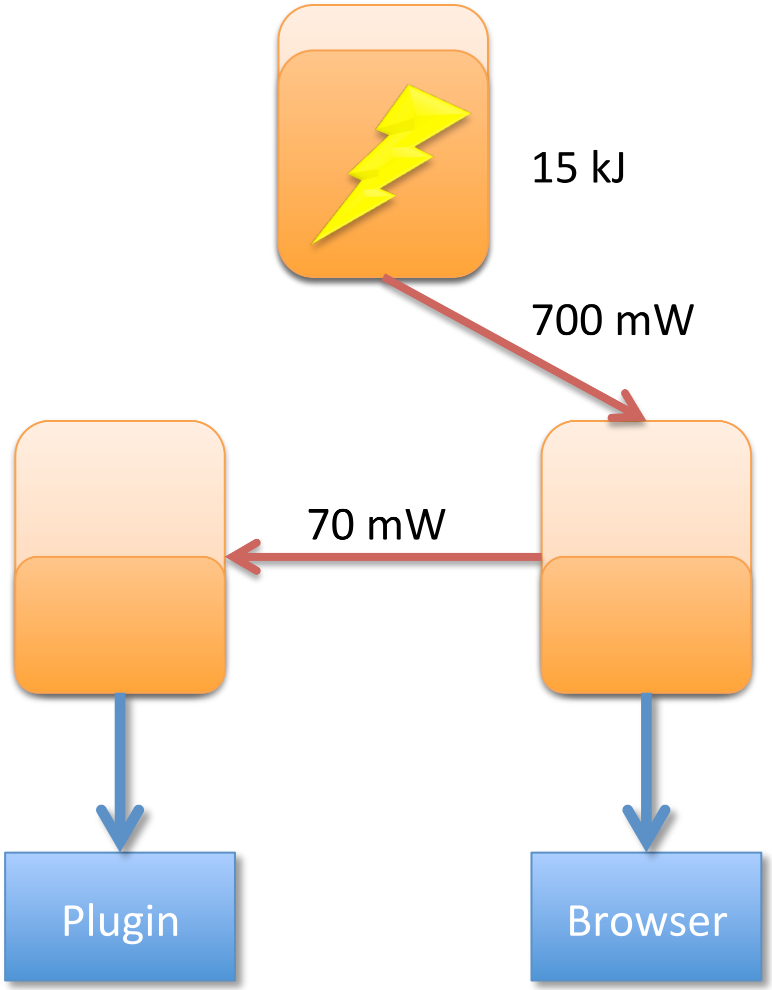
Taps

- A rate transfer between reserves
- Allows hierarchies



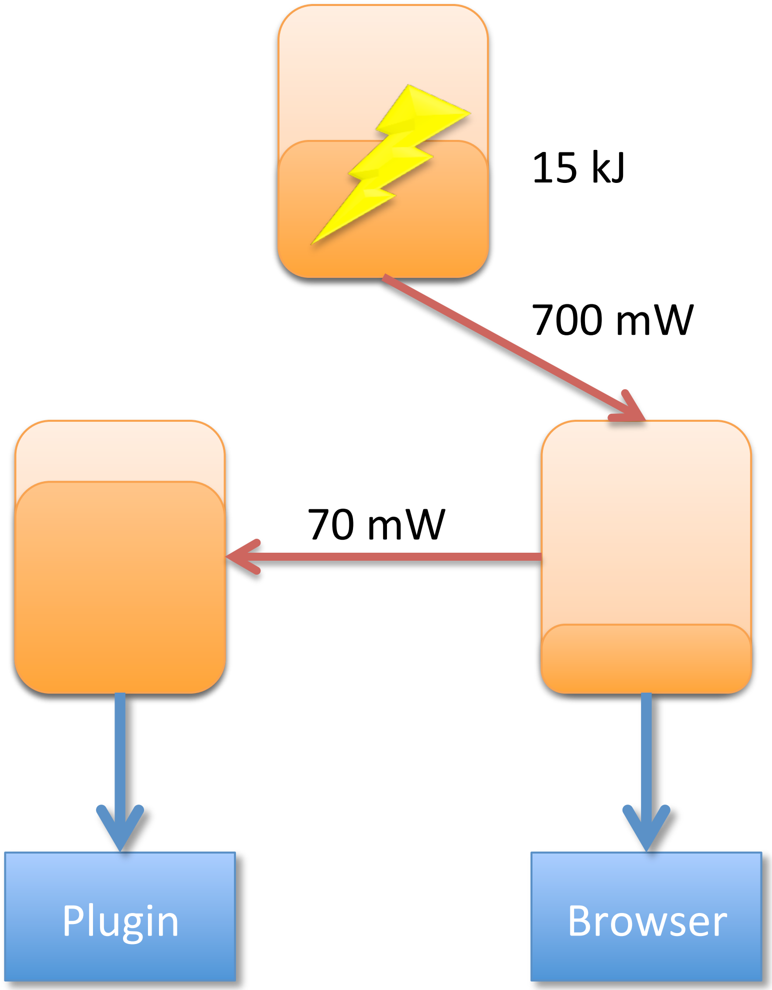
Taps

- A rate transfer between reserves
- Allows hierarchies



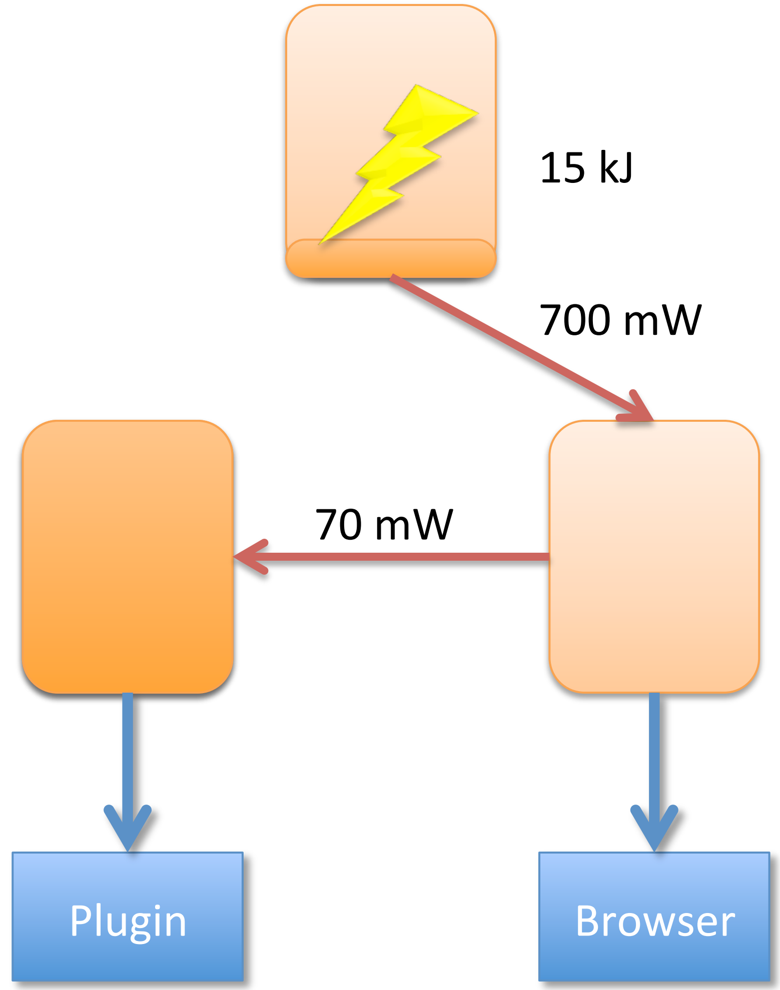
Taps

- A rate transfer between reserves
- Allows hierarchies



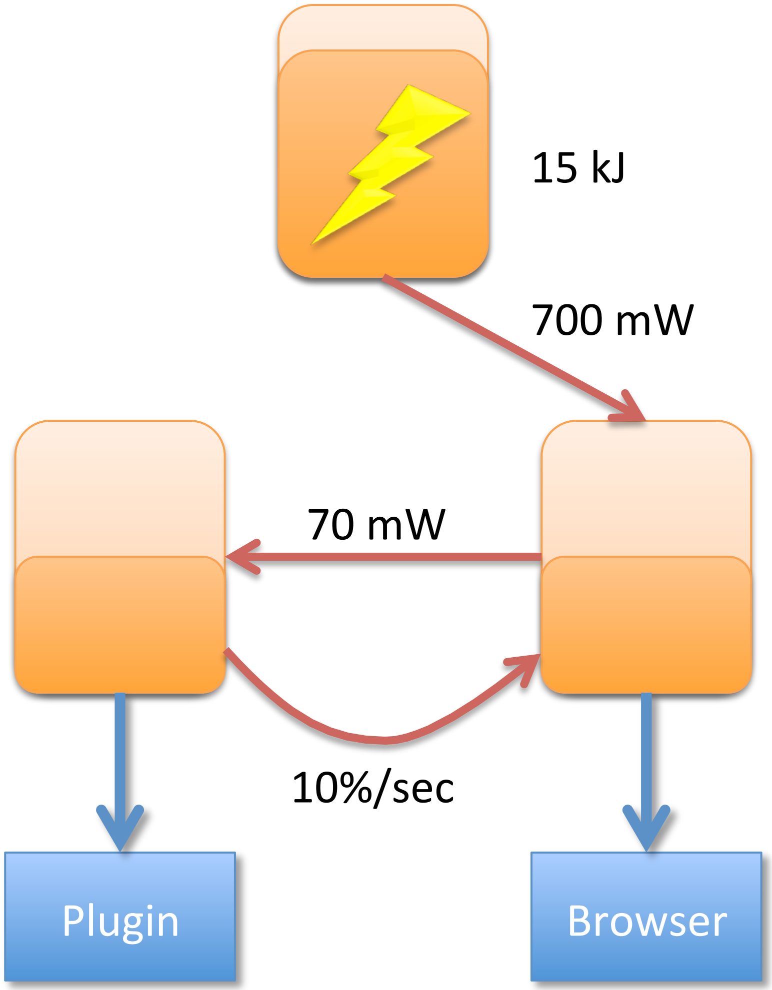
Taps

- A rate transfer between reserves
- Allows hierarchies



Taps

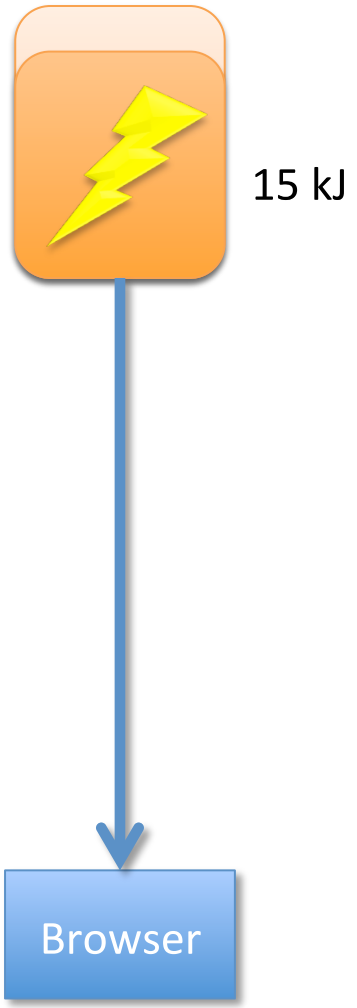
- A rate transfer between reserves
- Allows hierarchies
- Backward taps prevent hoarding



Battery

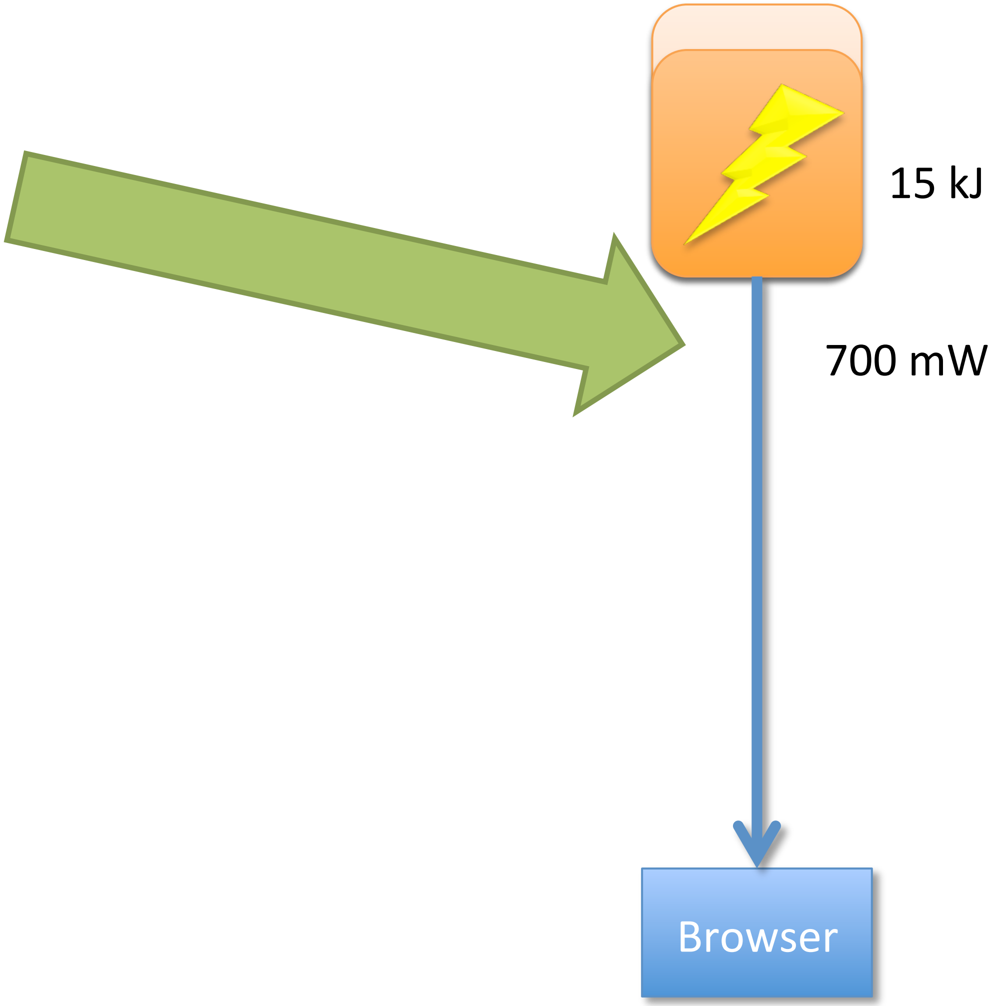
15 kJ

Consumption



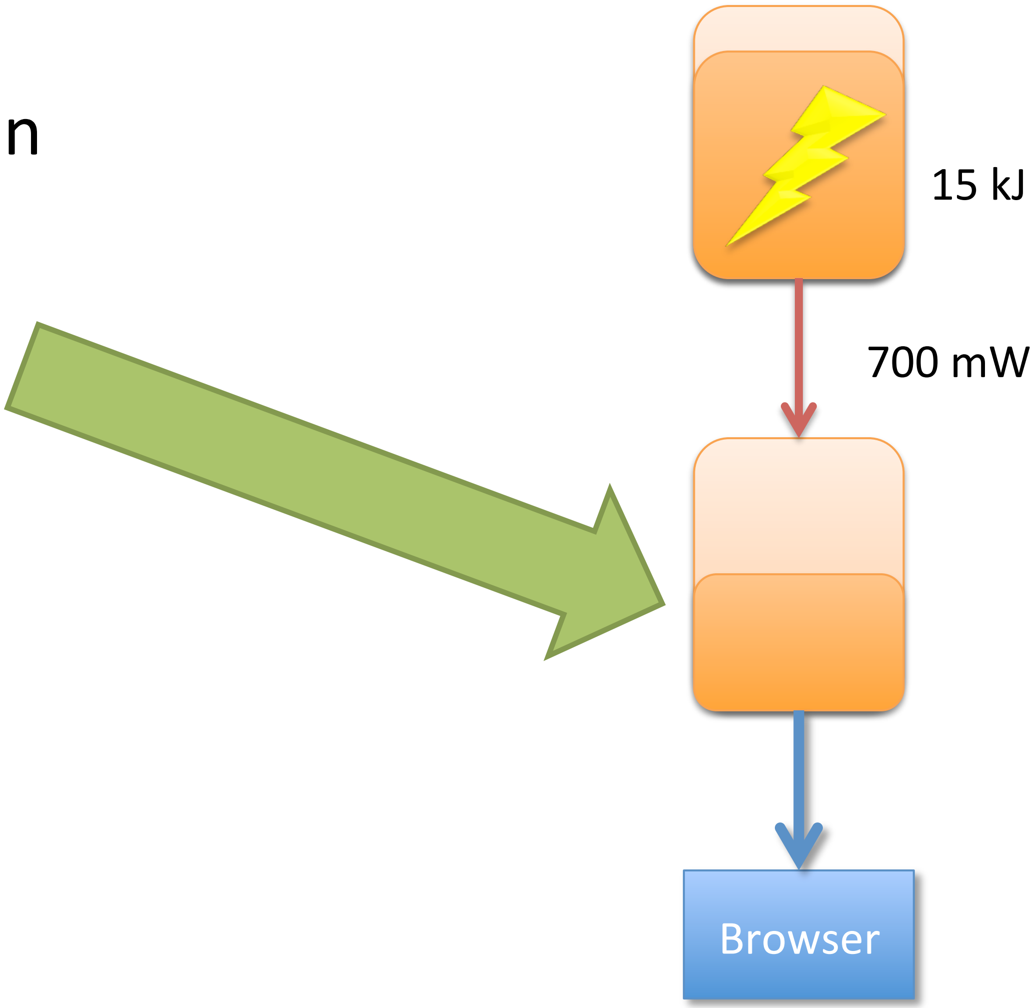
Throttling

- Throttle consumption
  - Taps



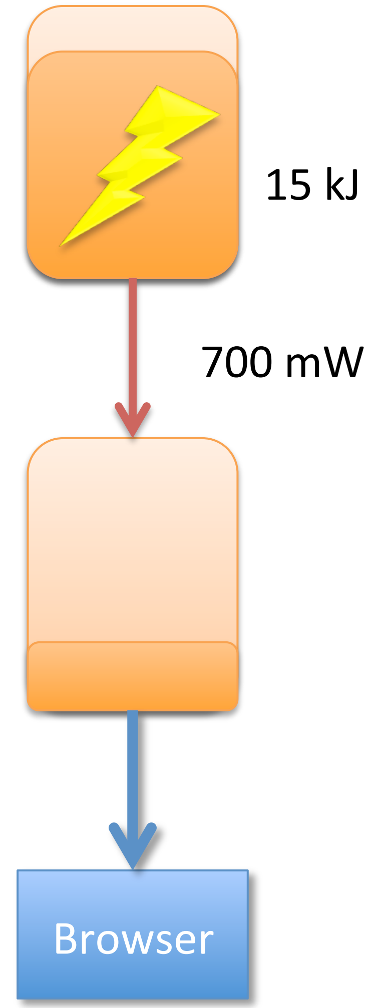
Accommodate Bursty Apps

- Throttle consumption
  - Taps
- Allow bursty apps
  - Reserves



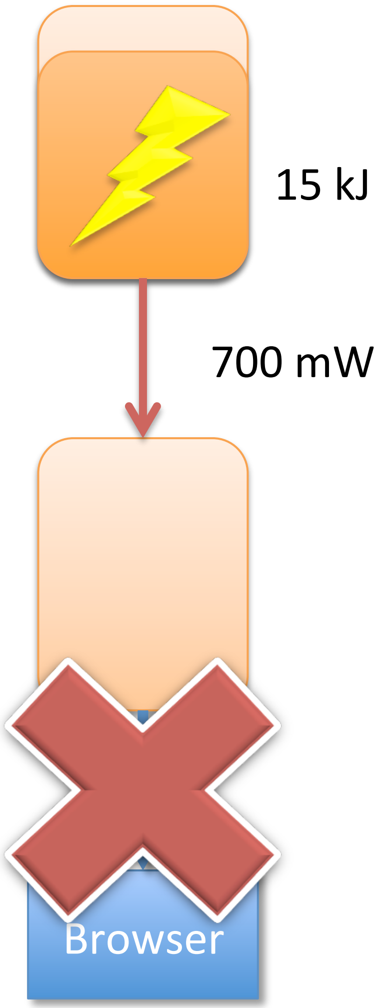
Accommodate Bursty Apps

- Throttle consumption
  - Taps
- Allow bursty apps
  - Reserves



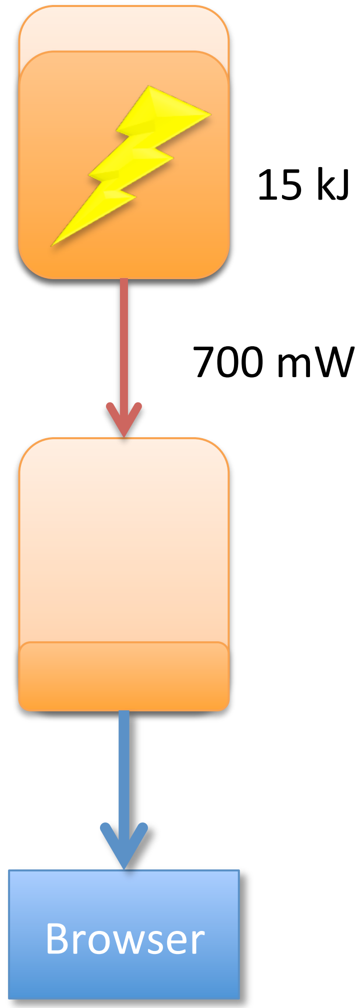
Accommodate Bursty Apps

- Throttle consumption
  - Taps
- Allow bursty apps
  - Reserves



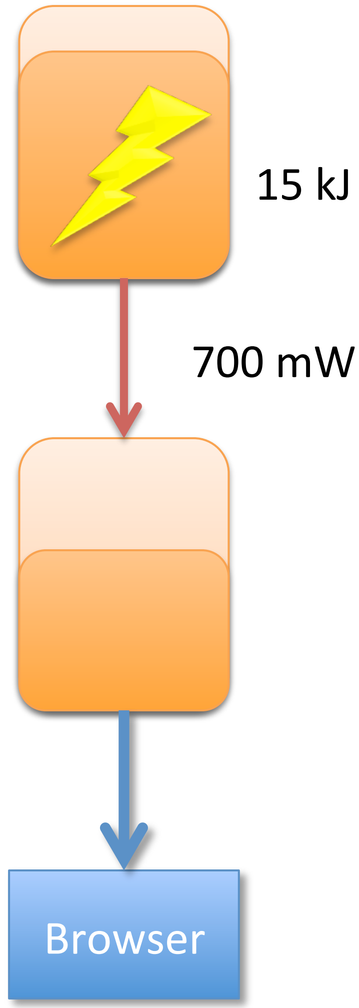
Accommodate Bursty Apps

- Throttle consumption
  - Taps
- Allow bursty apps
  - Reserves



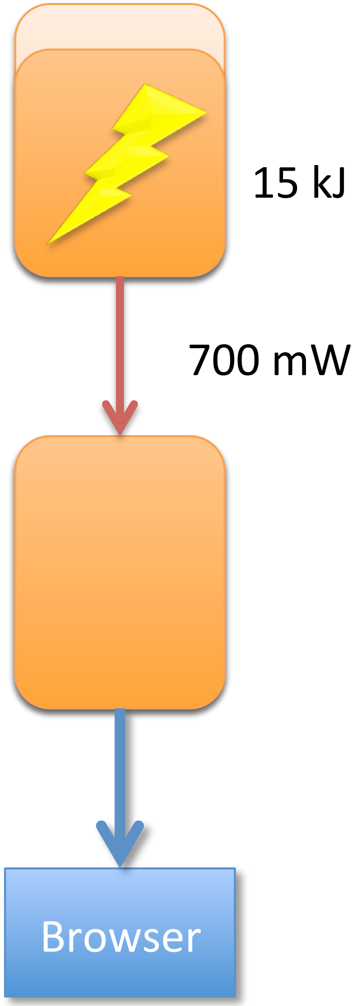
Accommodate Bursty Apps

- Throttle consumption
  - Taps
- Allow bursty apps
  - Reserves



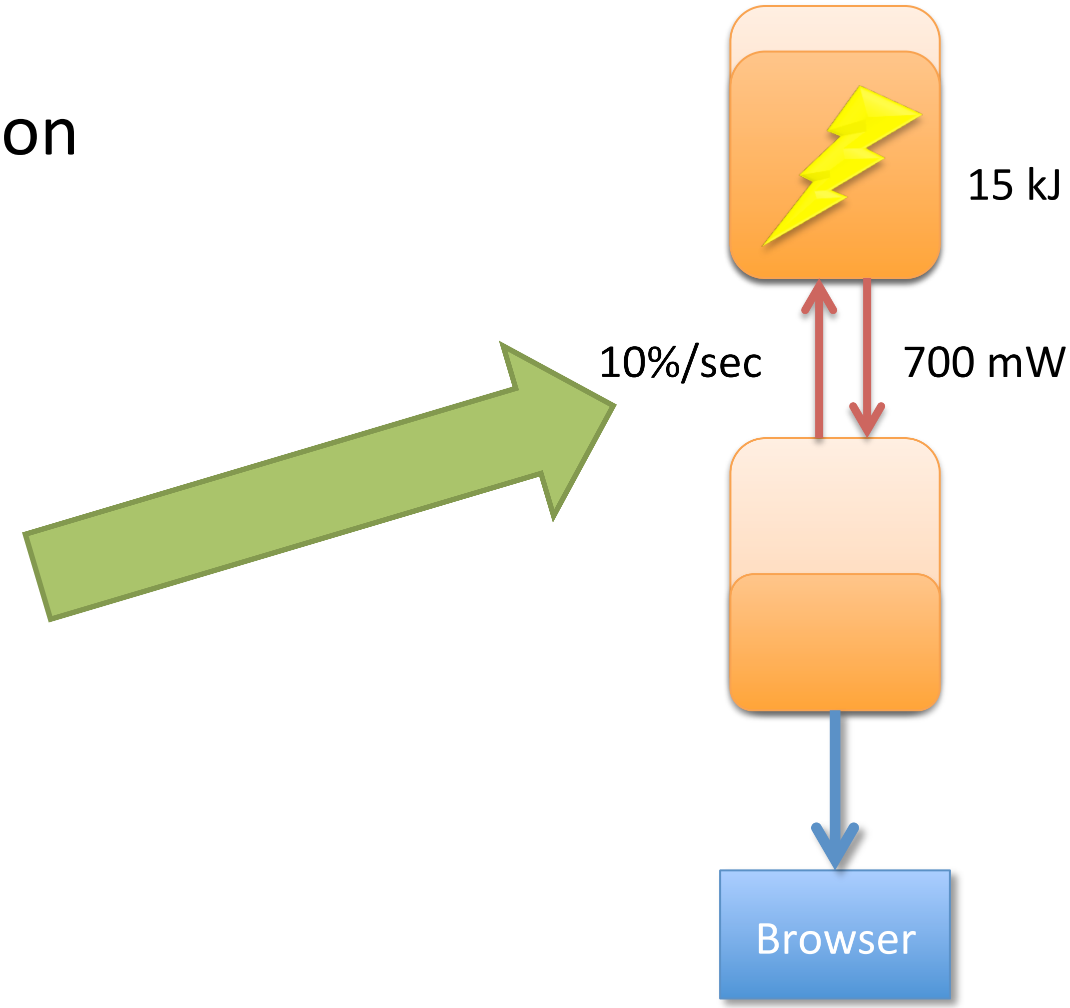
Accommodate Bursty Apps

- Throttle consumption
  - Taps
- Allow bursty apps
  - Reserves



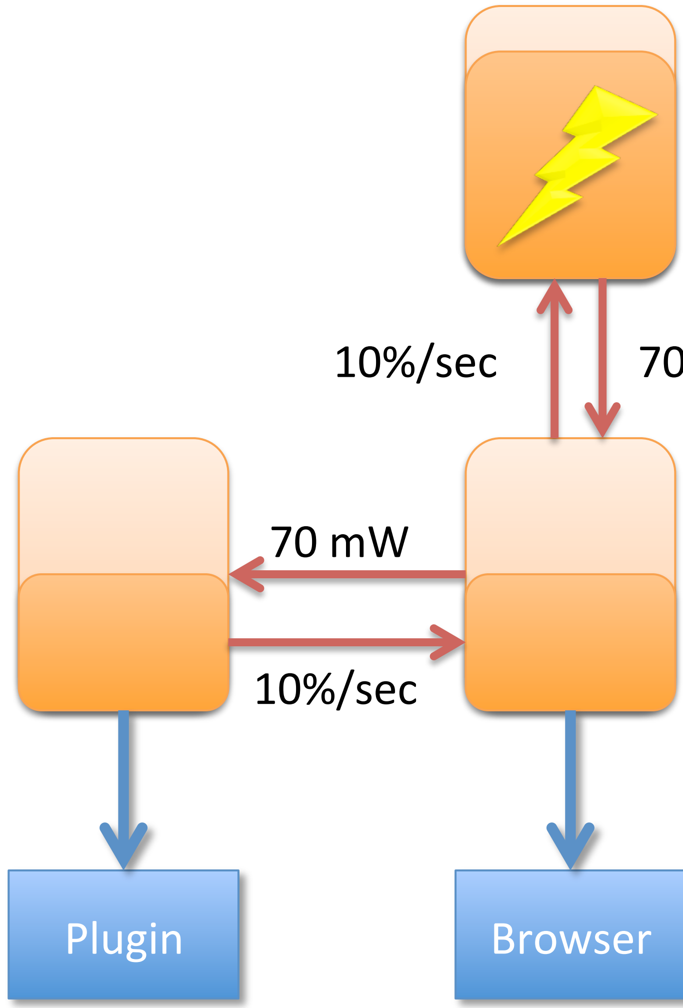
Prevent Hoarding

- Throttle consumption
  - Taps
- Allow bursty apps
  - Reserves
- Prevent hoarding
  - Backward Taps



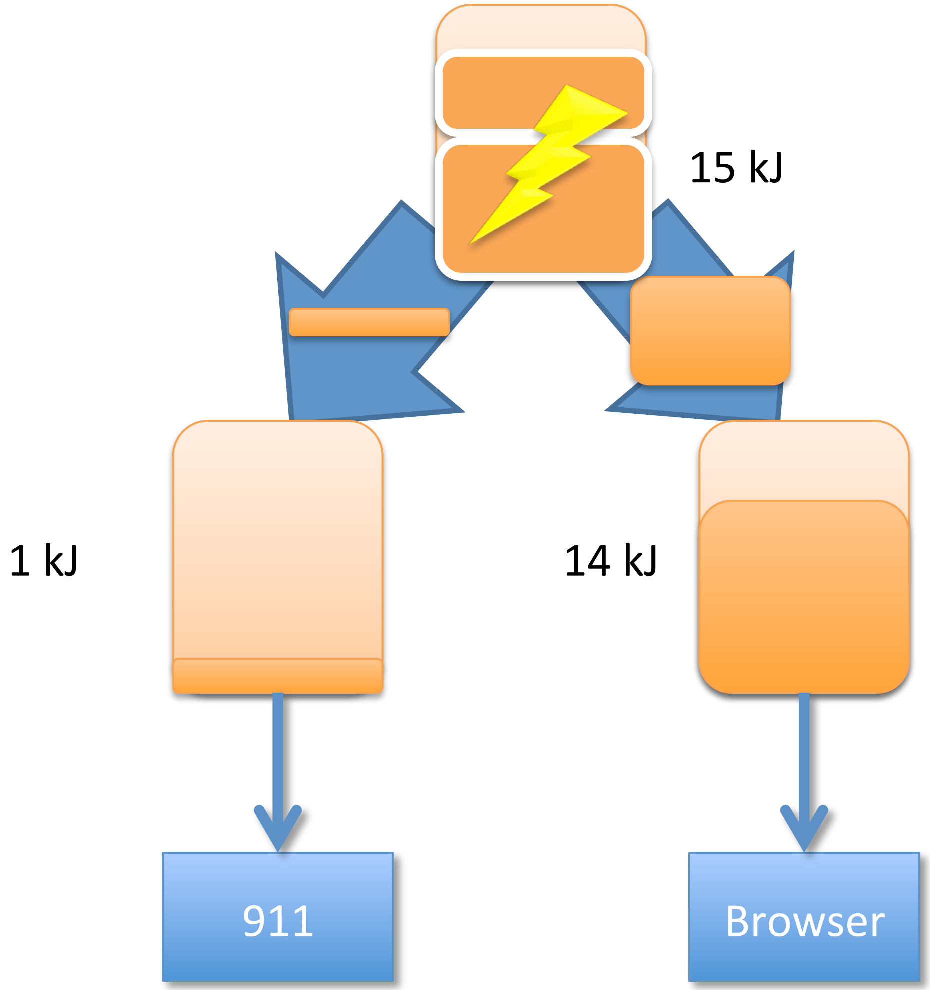
Fine-grained App Control

- Throttle consumption
  - Taps
- Allow bursty apps
  - Reserves
- Prevent hoarding
  - Backward Taps
- Fine-grained, app specific policies
  - Nesting



Subdivision & Isolation

- "Virtualized batteries"
- Guarantee energy to certain apps

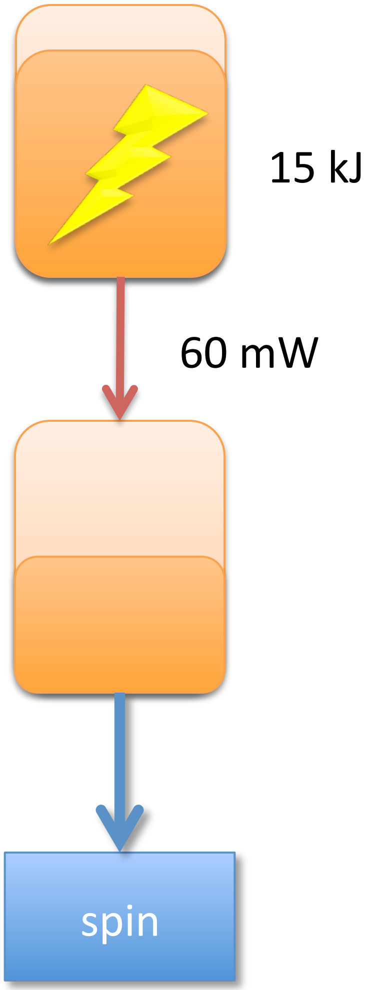


Outline

- New Abstractions for Control
- **Examples**
- The Problem of Closed Platforms
- Cinder-Linux

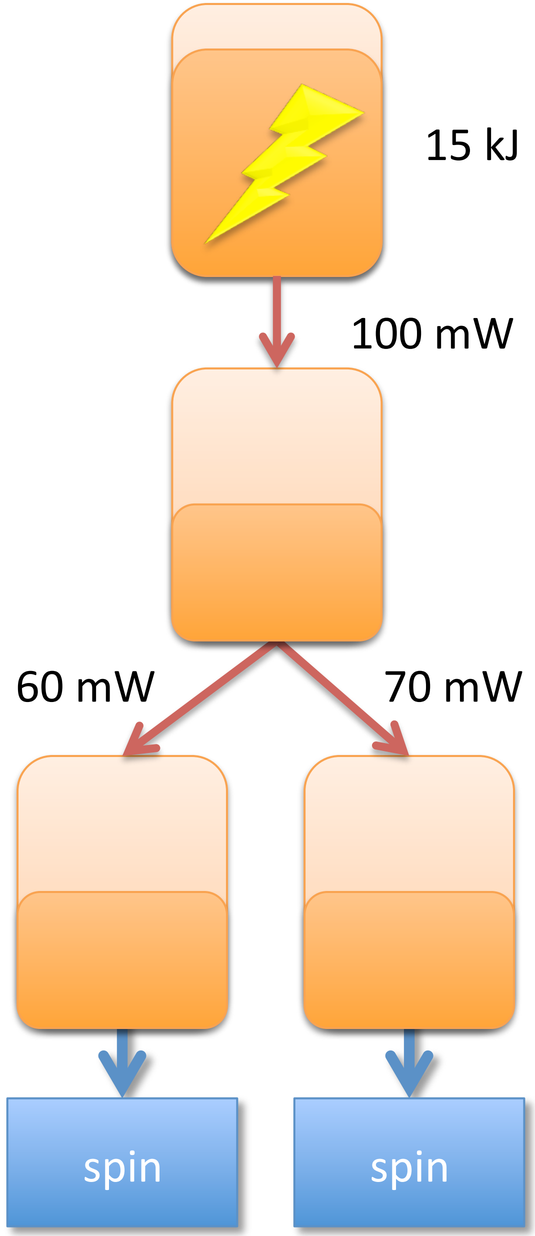
ewrap

```
ewrap 60 ./spin
```



ewrap

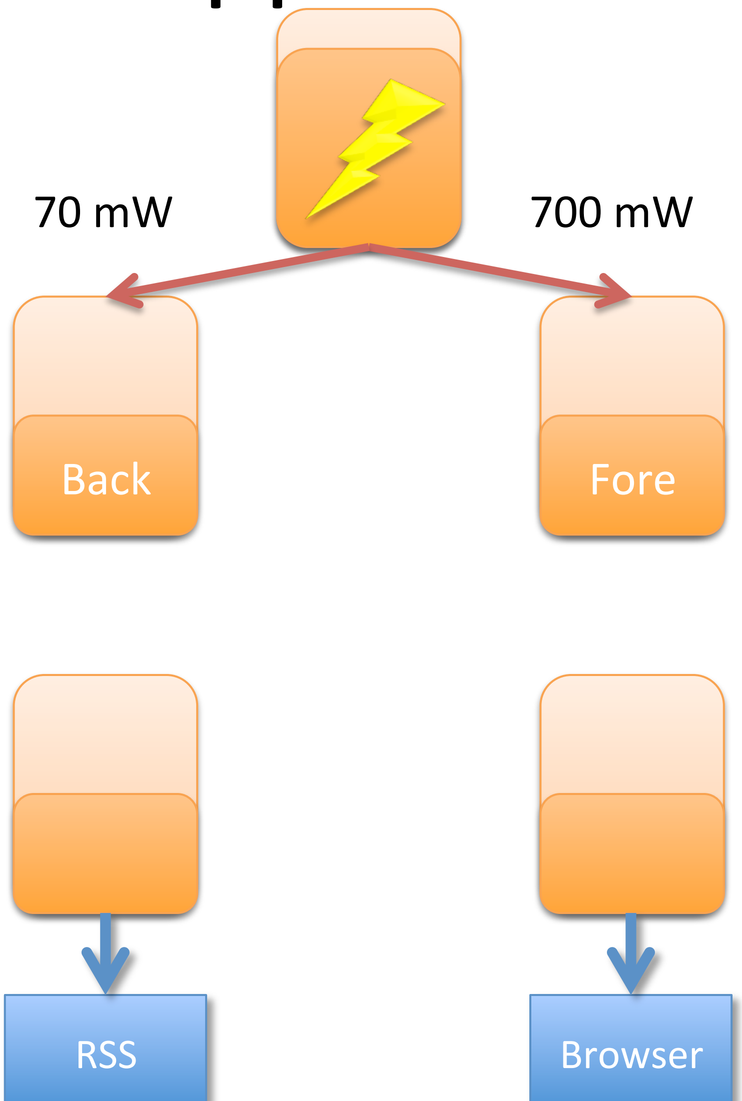
```
ewrap 100 bash -c \
 "ewrap 60 ./spin &
  ewrap 70 ./spin &"
```



Background Apps

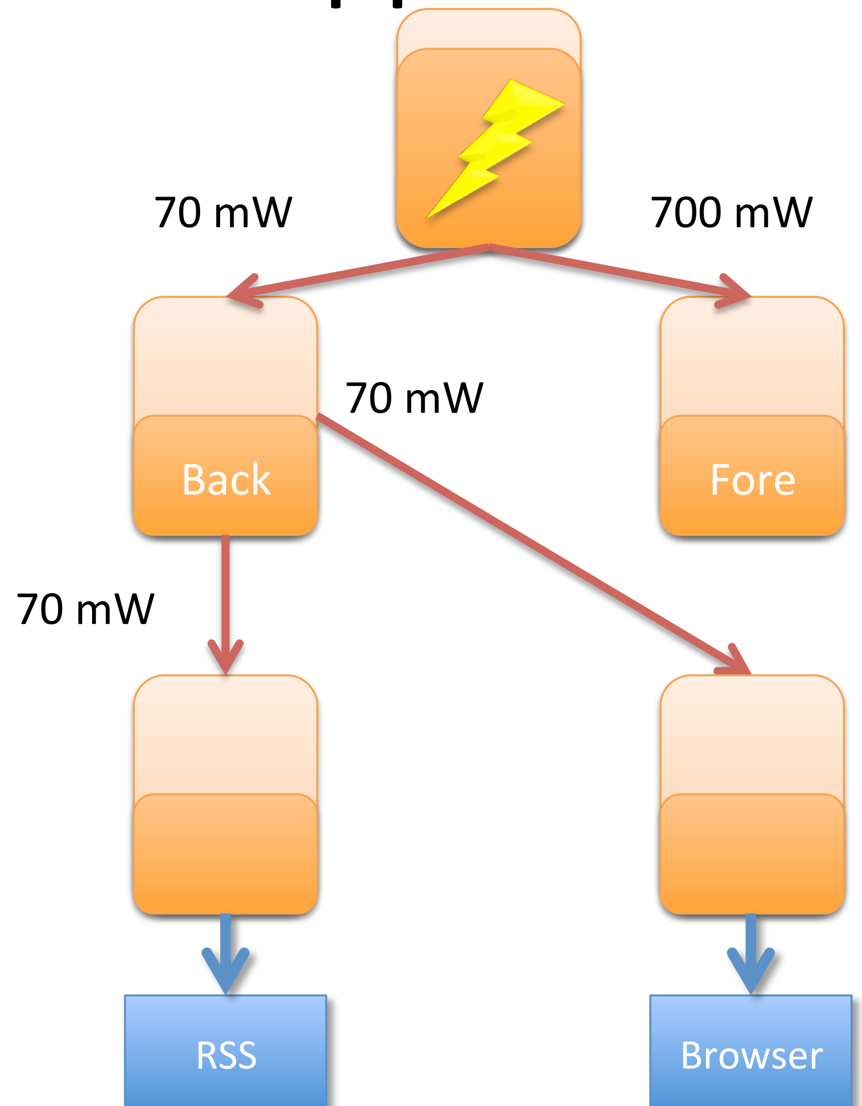
- Meet users' expectations

- Apps they can't see on screen shouldn't cause battery drain



Background Apps

- Meet users' expectations

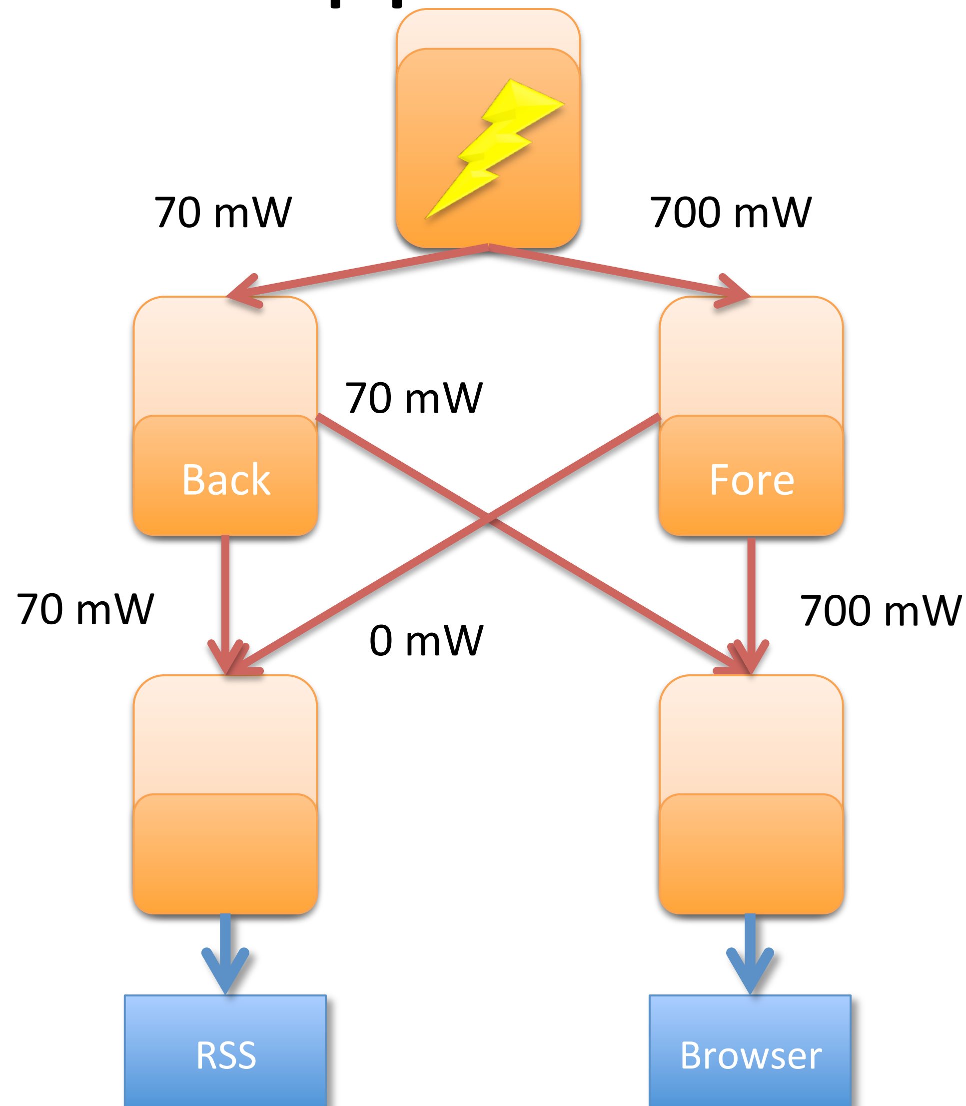


- Apps they can't see on screen shouldn't cause battery drain

Background Apps

- Meet users' expectations

- Apps they can't see on screen shouldn't cause battery drain

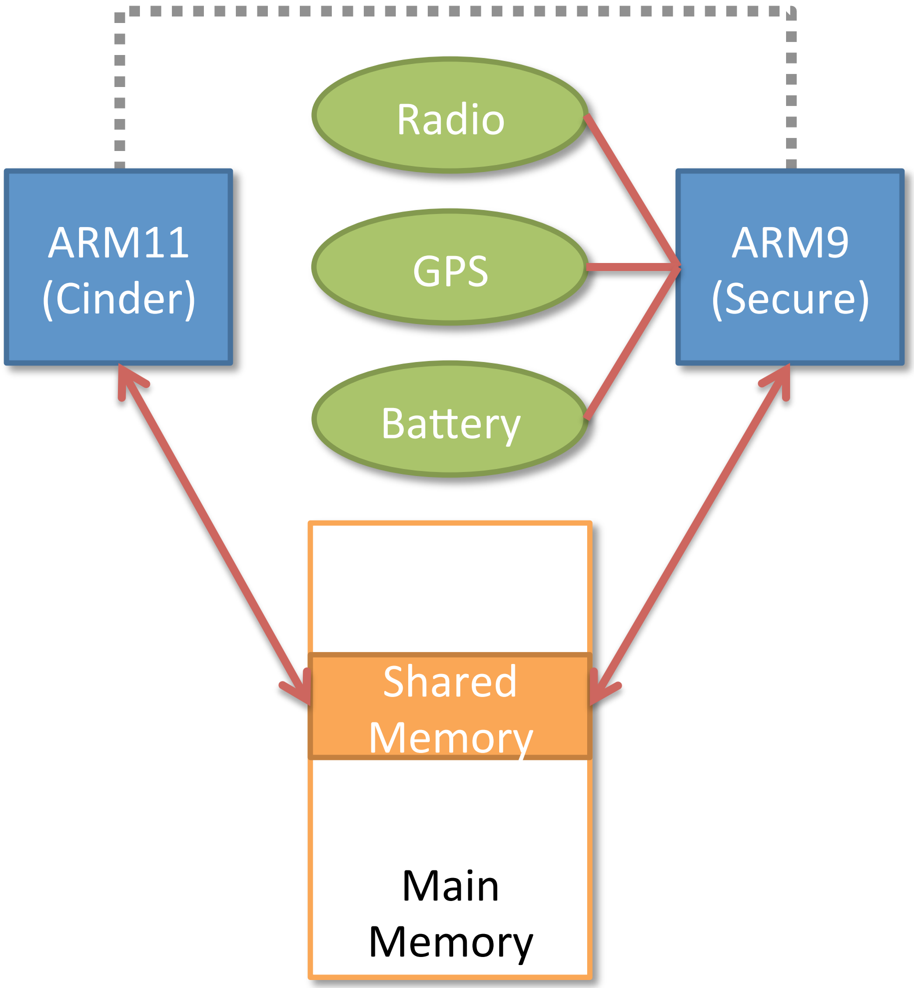


Outline

- New Abstractions for Control
- Examples
- **The Problem of Closed Platforms**
- Cinder-Linux

HTC Dream Design

- ARM11 CPU
  - Runs applications
- "Secure" ARM9
  - Controls devices
- Binary blob shared object to interact
- Difficult to reverse engineer the protocol



Developing for a Mobile Phone

- Getting worse
  - Locked bootloaders, software integrity checks
  - Research OSes on phones at all in the future?

- Evolution allowed, but not revolution
  - Can tweak apps, OS; cannot replace OS

- Dominant user computing platform
  - Systems research community is locked out

Outline

- New Abstractions for Control
- Examples
- The Problem of Closed Platforms
- **Cinder-Linux**

Cinder-Linux

- Implemented reserves and taps on Linux
  - Source code:
    http://www.scs.stanford.edu/histar/src

- Easier access to devices

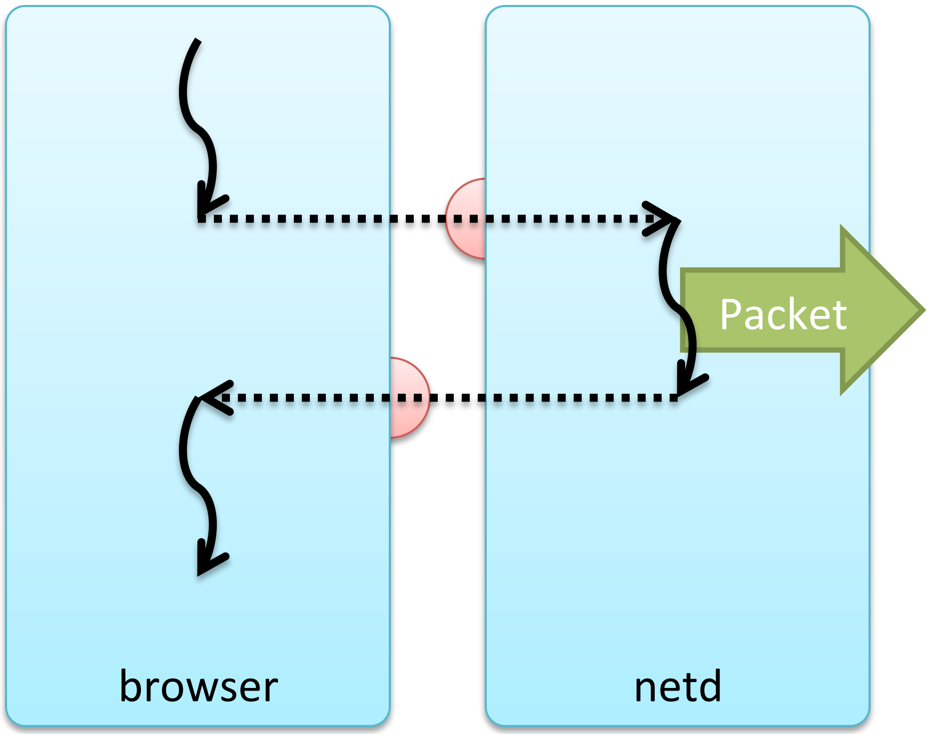
- Allow experimentation with more sophisticated workloads

The Problem with IPC

- Apps request service from daemons
- Daemons do work for the app
  - OS must bill the app correctly
- Not a problem on Cinder because of "gates"

Gates

- The basis for IPC
- A named entry point in an address space
- To request service, client threads enter the address space of a server
- Simplifies tracking



Cinder-Linux

- Must augment all IPCs across processes
  - Logic scattered throughout userspace

Conclusions

+ Users can control app energy use
+ Apps can leverage developer knowledge
+ Energy tracking works across boundaries
+ Easy to write “energy aware” applications

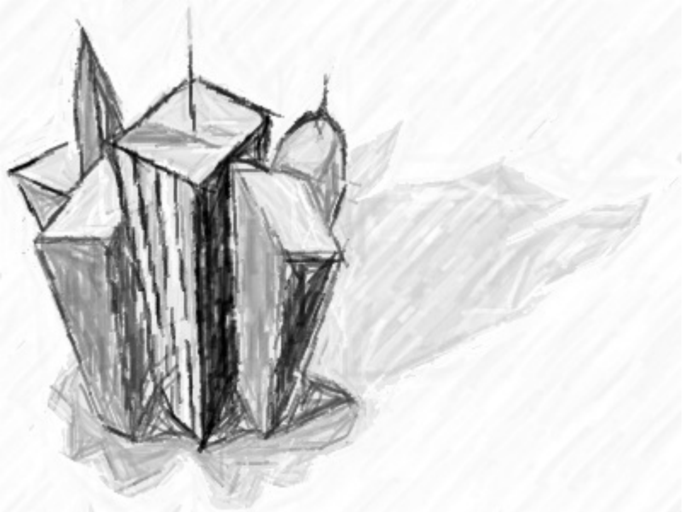
- Fighting manufacturer lock down
- Many knobs, may require sophistication

What's Next?

- Incorporate sophisticated energy modeling research, e.g. eprof
- Abstractions on Linux
  - Augment userspace code for energy tracking or do we need it?
    - eprof states most energy are consumed for accessing peripheral devices
- Android on Cinder

Mobile Apps: It's Time to Move Up to ConDOS

ConDOS: the Context Dataflow OS



david chu• aman kansal• jie liu• feng zhao◆
microsoft research redmond• microsoft research asia◆

APPS

How *Might* New Apps Use New Sensors?

camera

microphone (x2)

microphone

magnetometer

barometer

GPS

camera (x2)

accelerometers

light sensors

NFC

infrared camera

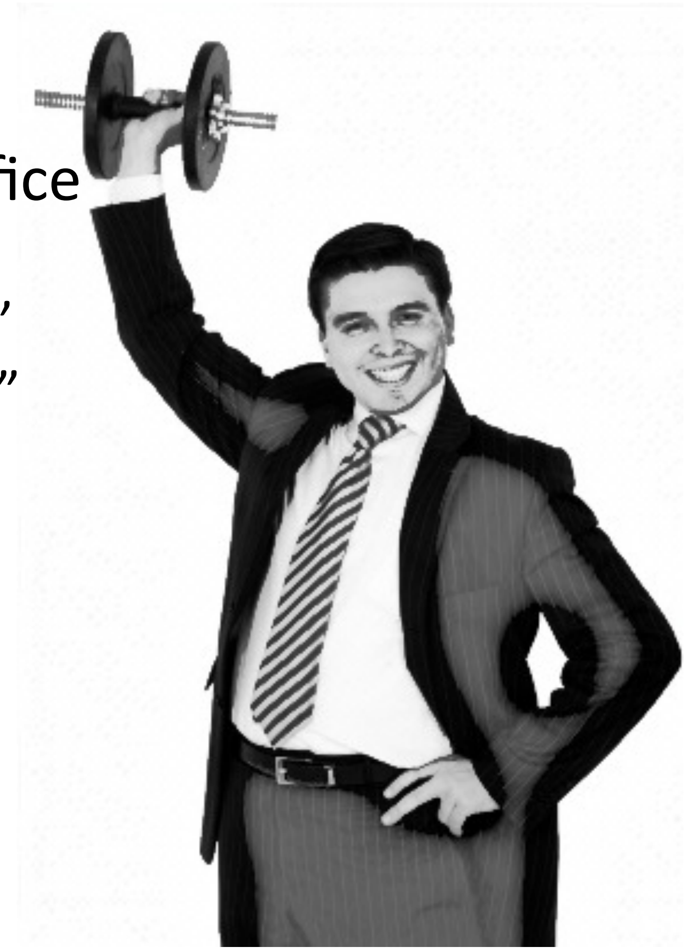
health sensors

gyroscopes

SENSORS

thermometer

OfficeFit app



- contextual fitness reminders in the office
  - "Don't slouch while sitting."
  - "You've been at your desk for too long."
  - "Take the stairs instead of the elevator."

- how it works
  - motion from IMU + sound from mic → various fitness activities
  - do this continuously

context data from sensors

- key pieces are ready
  - sensor hardware
  - application scenarios
  - algorithms (high accuracy inference, signal processing, db, etc.)

Motion State
sitting, walking, running

Logical Location
home, office, mall

Interruptible
yes, no

- where is the context?

- who is responsible for context?
  - individual apps
    - … but mobile OSs limit apps to foreground
    - … or apps can run anything in the background(!)
  - the cloud
    - … but energy cost of TX/RX is high
  - the mobile OS

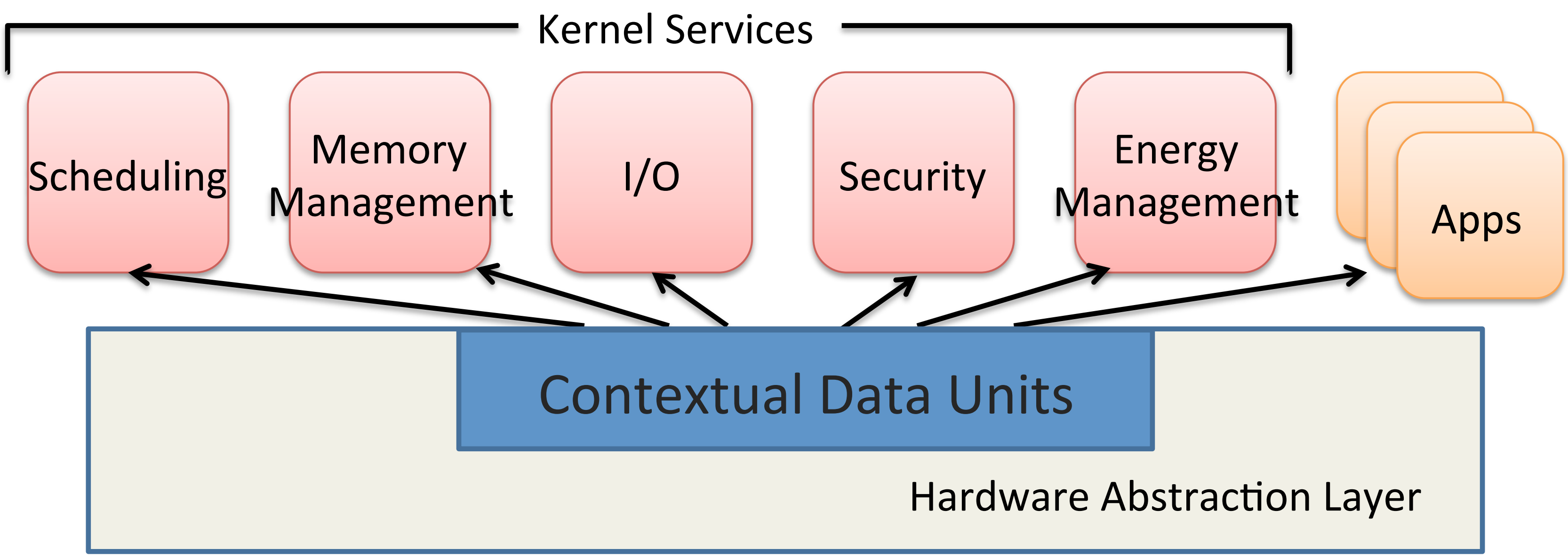
ConDOS design proposal

- export *Context Data Units (CDUs)* rather than raw sensor data
  - higher-level abstraction than bytes
  - apps query or subscribe to CDUs
- each CDU is defined by a CDU Generator: a graph of processing components
  - combine Generators into composite context dataflow (like packet dataflow [kohler '00])
  - provide a base CDU vocabulary (that is extensible)

Logical Location	Motion State	Interruptible
home, office, mall	*sitting, walking, running*	*yes, no*

benefits of OS-managed context

1. System services can use context



system services can use context

- memory management
  - preload calendar, email when *in the office*

Context	Preload Action
in the office	Email, Calendar
at a party	Twitter, Facebook

Memory Management

- I/O
  - ring volume adjusted based on *conversation*
  - networking params dictated by *movement* [Balakrishnan '10]

I/O

system services can use context

- security
  - auto password unlock when at *home*
  - lend your phone to others easily [liu '09]

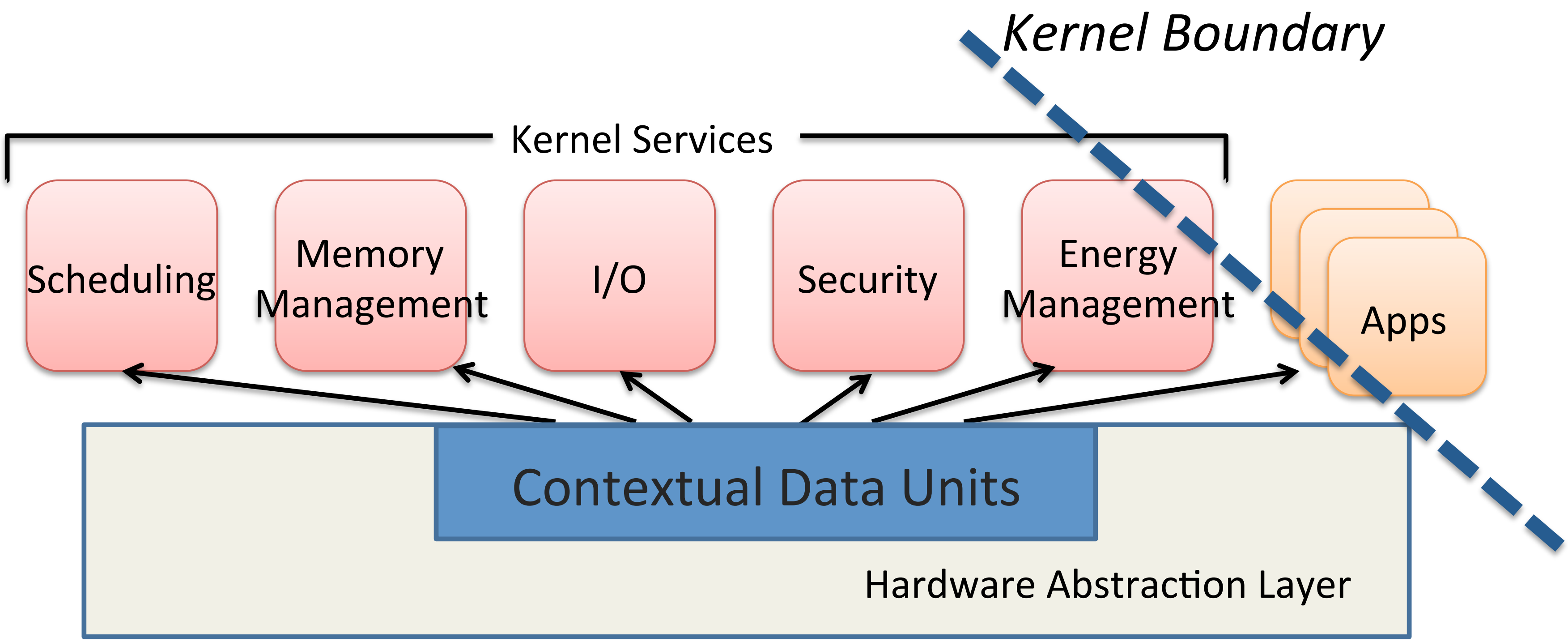
Security

- energy management
  - predict time-to-recharge based on context

Energy Management

benefits of OS-managed context

2. Privacy enforced by OS protection

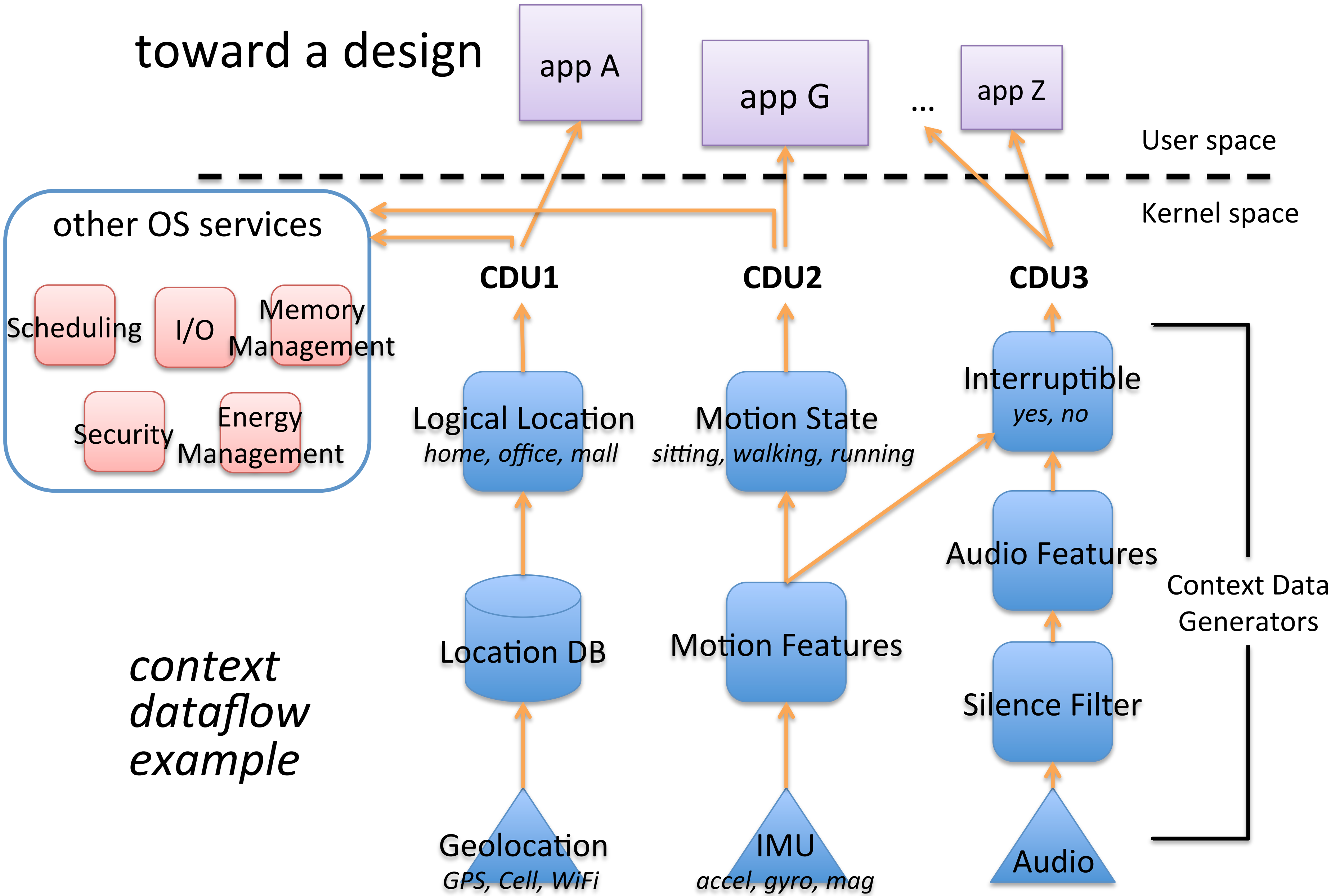


better sensor privacy

- mobile privacy is under attack [TaintDroid]
  - protecting raw sensor data is "trust the EULA"
  - 2/3 of popular apps use your data suspiciously
  - no idea what your raw data might be used for
- OS-managed context lets us do better
  - app install time: per CDU type access control
    - … vs. per sensor type access control
  - app run time: visual inspection of CDUs shared [Howell '10]
    - … vs. no comprehension of what is being shared
  - enforcement is low overhead

Courtesy D. Chu et. al.

toward a design



context dataflow example

Courtesy D. Chu et. al.

summary

- mobile OSs that don't *make* sense make *no* sense
  - *ConDOS* offers context as a primary app-OS interface
- apps, OS services and User Privacy may all benefit

Closing Thoughts

- Reserve and tap in Cinder enable fine grained control on application energy consumption.
  – But there is no easy answer on when to use them?
- When to generate contextual information?
  – A high overhead is incurred to acquire it.
- What else need OS support?