Enhancing Mobile Apps to Use Sensor Hubs without Programmer Effort

Haichen Shen, Aruna Balasubramanian, Anthony LaMarca, David Wetherall

Continuous sensing apps

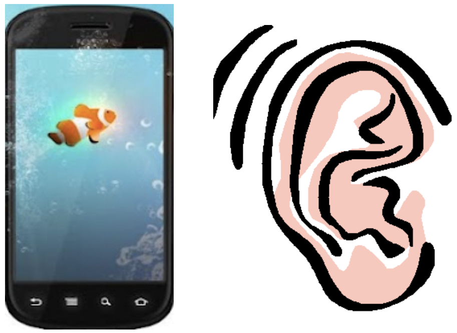
Step Counting

Fall Detection

Driver Monitor

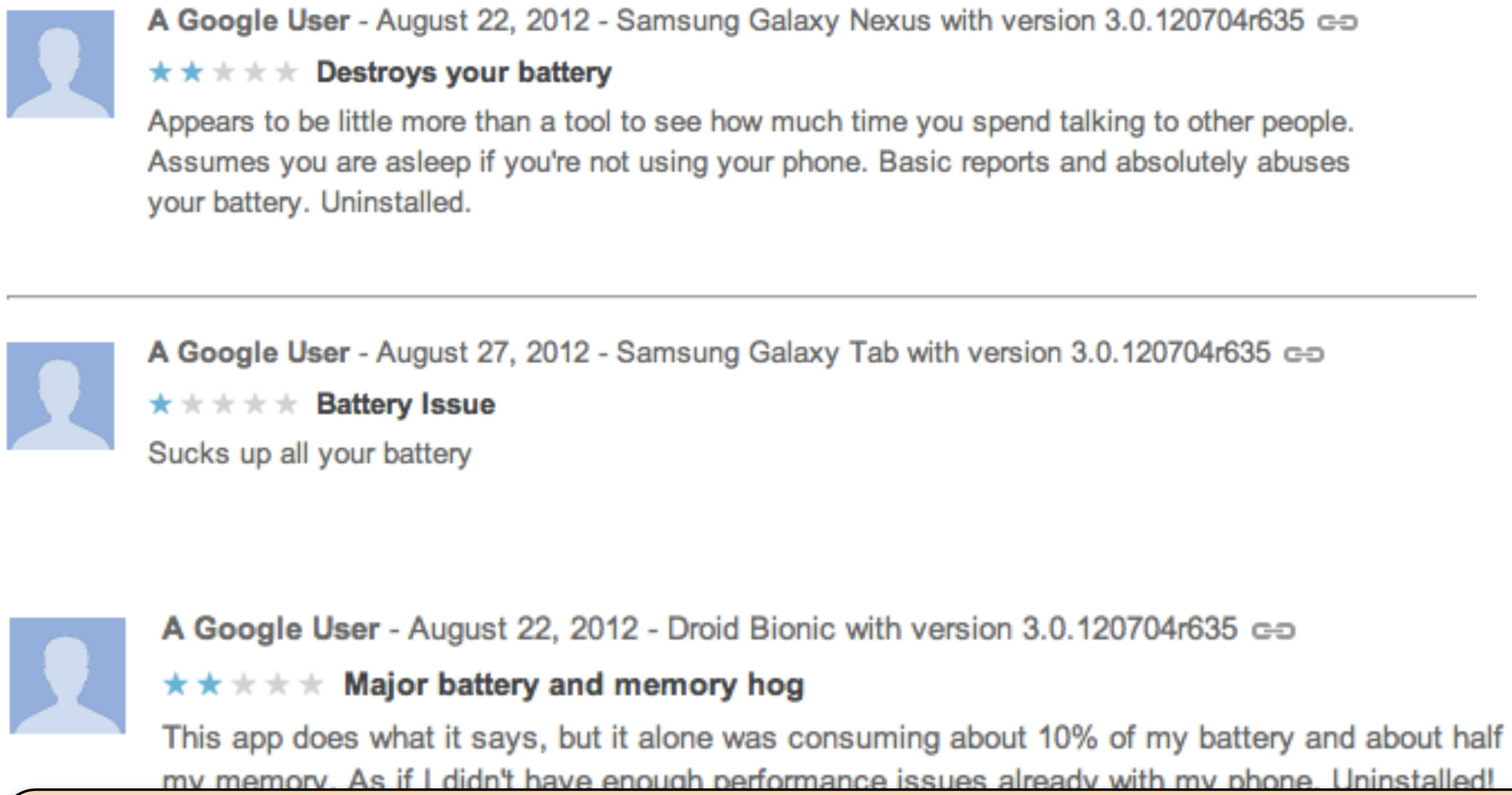
Theft Detection

Healthcare apps:
Ambulation


Lifestyle monitoring:
BeWell, Acoustic

Participatory sensing:
MobiPerf

But it drains the battery



A Google User - August 22, 2012 - Samsung Galaxy Nexus with version 3.0.120704r635

★★☆☆☆ **Destroys your battery**

Appears to be little more than a tool to see how much time you spend talking to other people. Assumes you are asleep if you're not using your phone. Basic reports and absolutely abuses your battery. Uninstalled.

A Google User - August 27, 2012 - Samsung Galaxy Tab with version 3.0.120704r635

★☆☆☆☆ **Battery Issue**

Sucks up all your battery

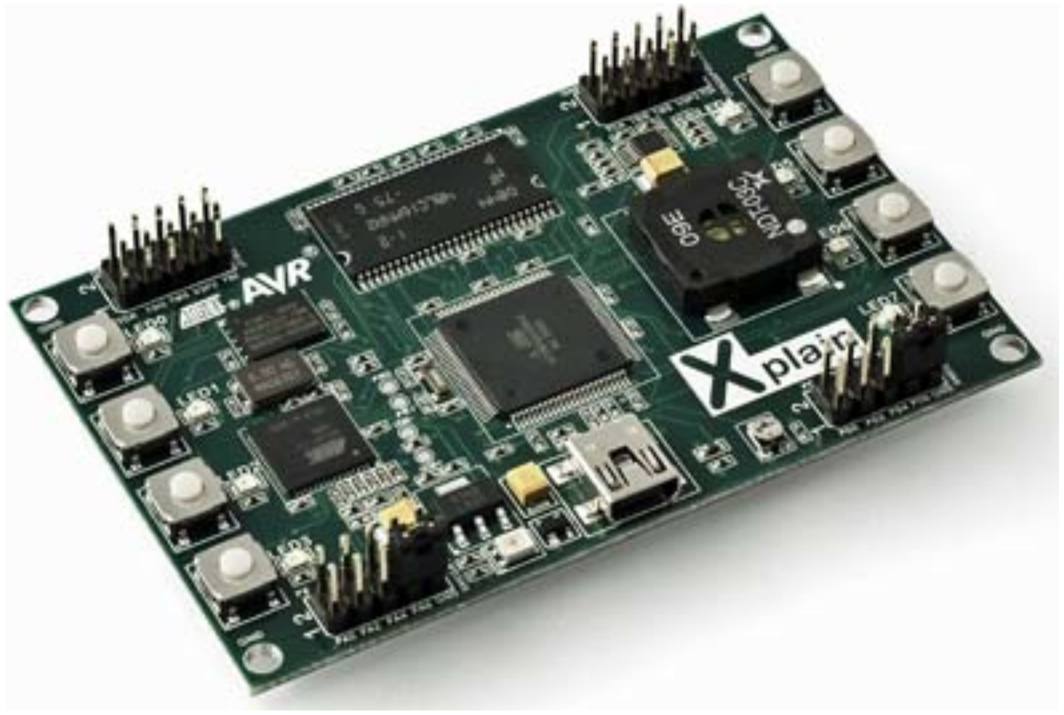
A Google User - August 22, 2012 - Droid Bionic with version 3.0.120704r635

★★☆☆☆ **Major battery and memory hog**

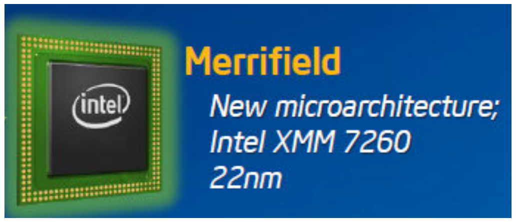
This app does what it says, but it alone was consuming about 10% of my battery and about half my memory. As if I didn't have enough performance issues already with my phone. Uninstalled!

Problem: CPU frequently wakes up to process sensor data

Sensor hub: low power processor



~1.5 mW at 2MHz



Intel Merrifield

TI's Tiva

Apple M7

Existing approaches make it hard to leverage sensor hub

APIs

- Provided by software company, e.g. Apple, Google

✓ Easy to program

✗ Only support a set of pre-defined events

✗ Require programmer effort

Hardware SDK

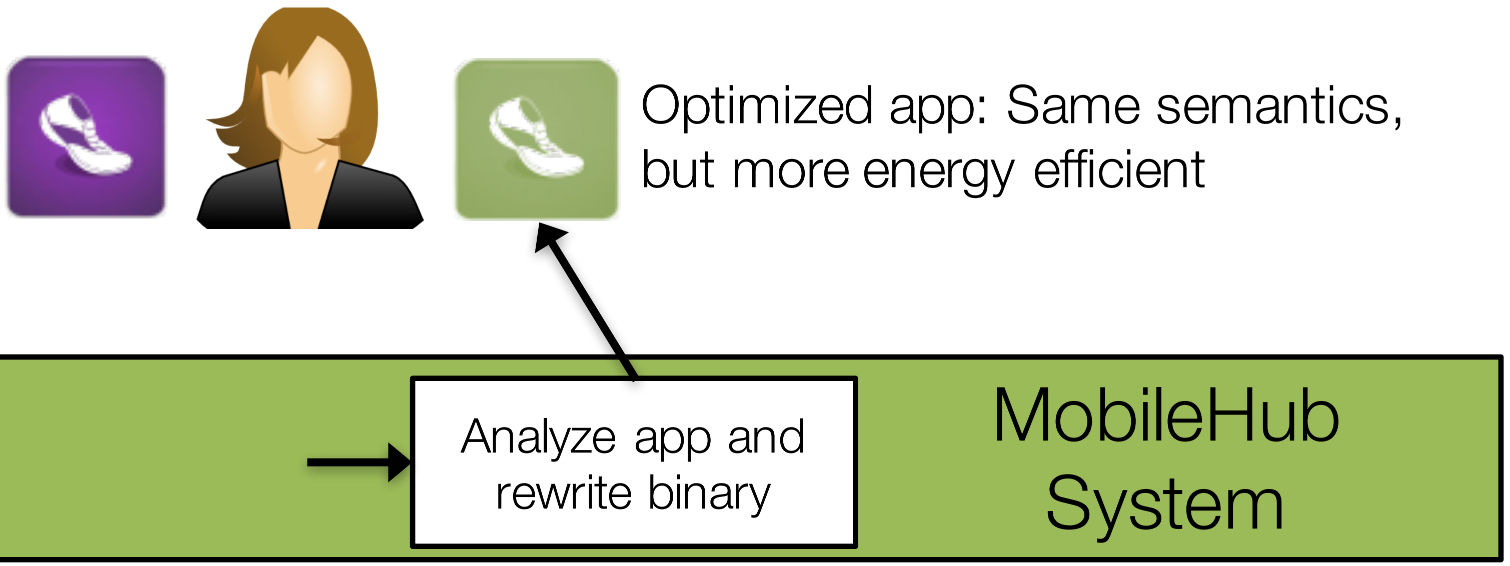
- Provided by hardware manufacturer, e.g. TI TivaWare

✓ Full control of sensor hub

✗ Compatibility

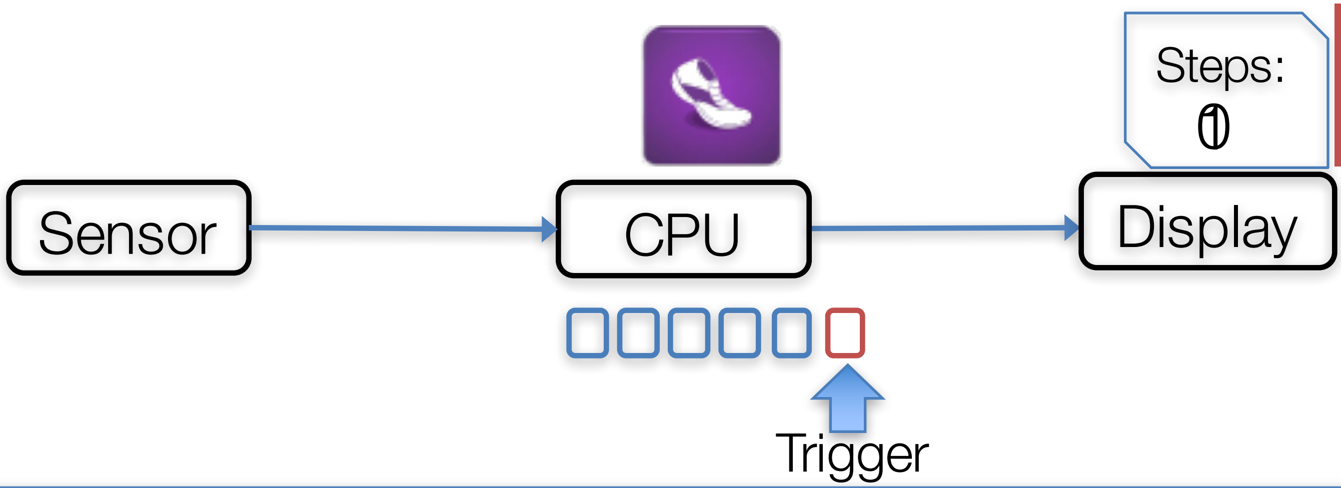
✗ Require programmer effort

MobileHub: leverage sensor hub without programmer effort



Optimized app: Same semantics, but more energy efficient

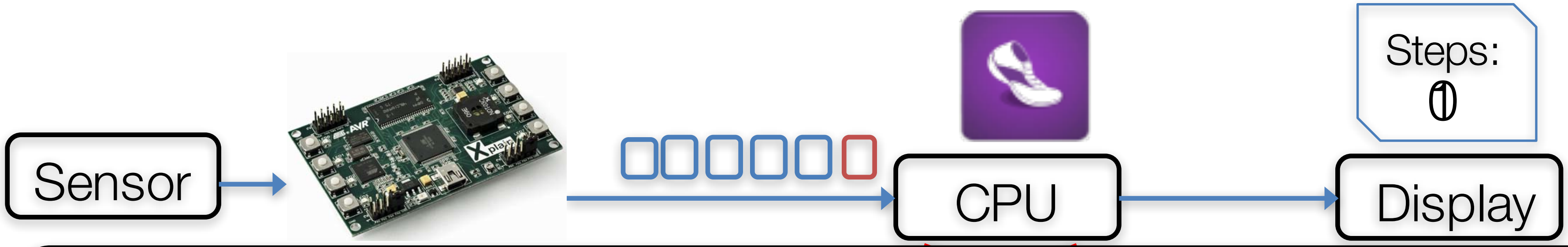
MobileHub example



App Notification

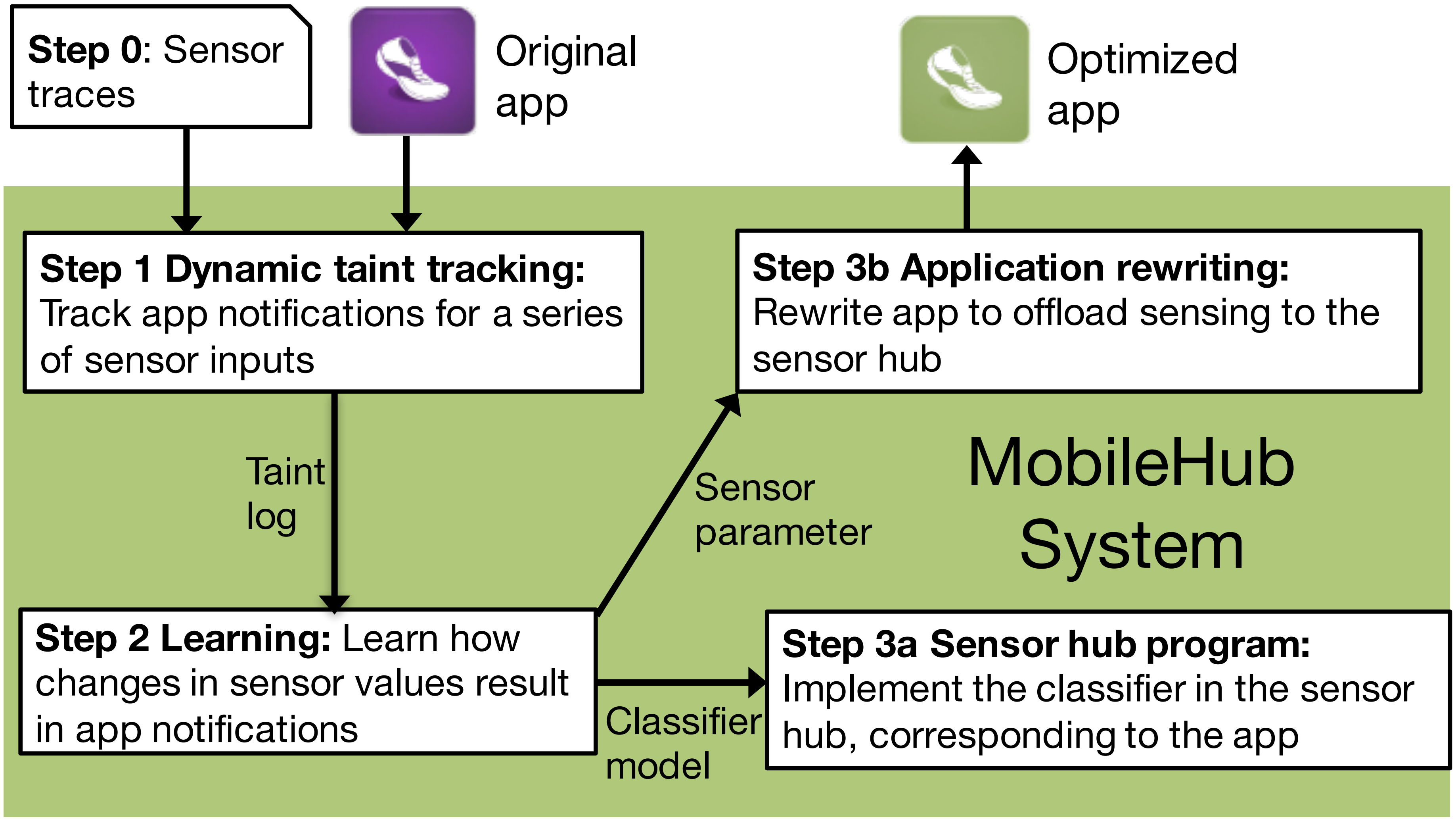
MobileHub

No change to app semantics

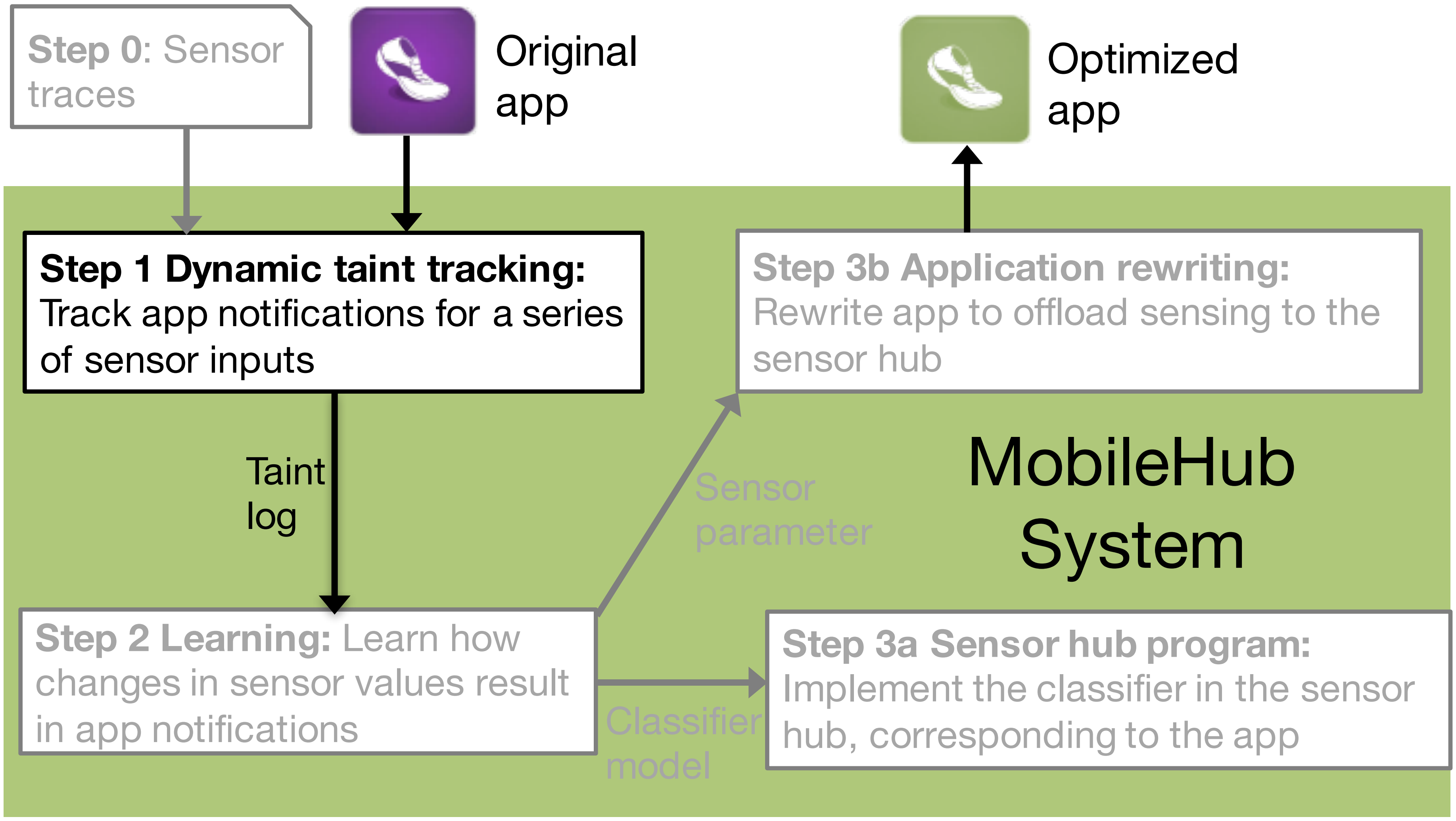


Challenge: How does MobileHub know when the application needs to be triggered?

MobileHub system overview


Step 0: Sensor traces
Original app
Optimized app
Step 1 Dynamic taint tracking: Track app notifications for a series of sensor inputs
Step 3b Application rewriting: Rewrite app to offload sensing to the sensor hub
Taint log
Sensor parameter
MobileHub System
Step 2 Learning: Learn how changes in sensor values result in app notifications
Classifier model
Step 3a Sensor hub program: Implement the classifier in the sensor hub, corresponding to the app

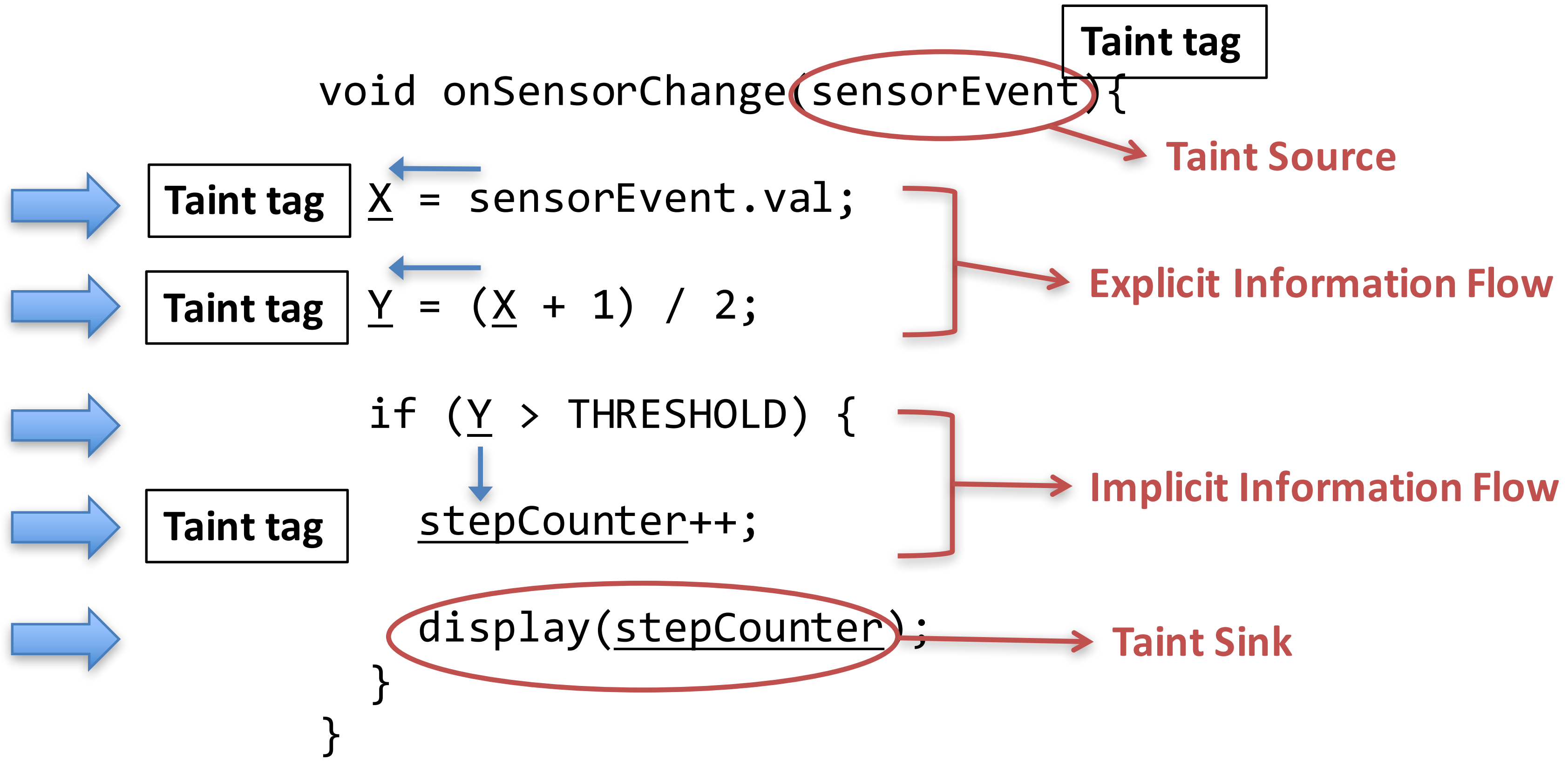
MobileHub system overview


Step 0: Sensor traces
Original app
Optimized app
Step 1 Dynamic taint tracking: Track app notifications for a series of sensor inputs
Taint log
Step 2 Learning: Learn how changes in sensor values result in app notifications
Sensor parameter
Classifier model
Step 3b Application rewriting: Rewrite app to offload sensing to the sensor hub
Step 3a Sensor hub program: Implement the classifier in the sensor hub, corresponding to the app
MobileHub System

Why do we need taint tracking?

- Goal: to track when a sensor value leads to an app notification.
- Observing the app notifications alone is insufficient.
- Use taint tracking to track the sensor data from when it was recorded to when it was used by the application

Taint tracking example



```
void onSensorChange(sensorEvent){
    X = sensorEvent.val;
    Y = (X + 1) / 2;
    if (Y > THRESHOLD) {
        stepCounter++;
        display(stepCounter);
    }
}
```

Challenge: implicit flow tracking

- Most taint tracking platforms only track explicit flow
- Without implicit flow tracking, we could only track 20% of triggers for sensing apps
- Use instrumentation to force implicit flow tracking
    - Built on top of TaintDroid [Enck_OSDI2010]

Instrumentation for implicit flow tracking

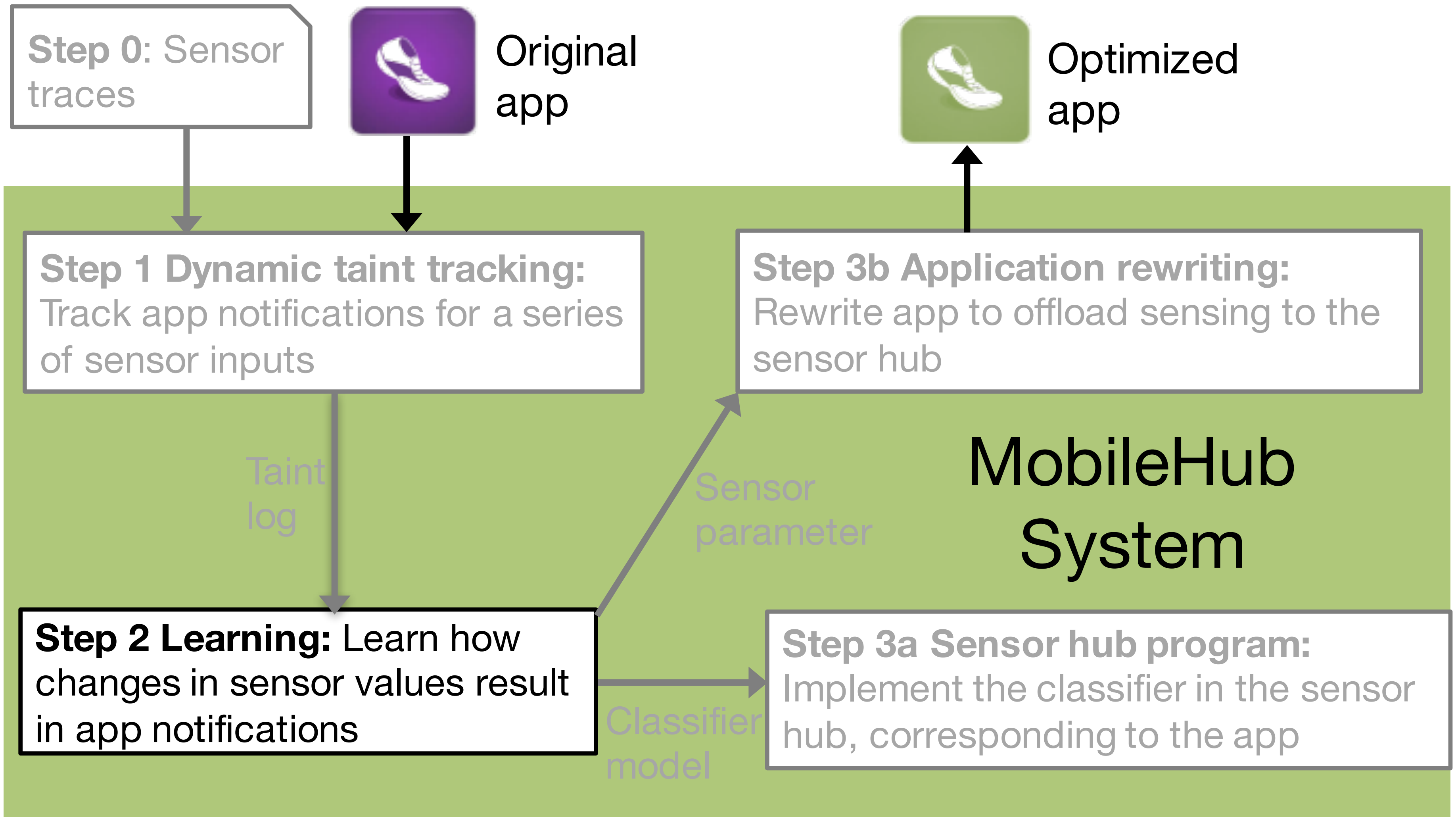
```
    void onSensorChange(sensorEvent){

Taint tag   X = sensorEvent.val;

Taint tag   Y = (X + 1) / 2;
            tag = getTaintTag(Y);
            if (Y > THRESHOLD) {

              stepCounter++;
              Taint(stepCounter, tag);
              display(stepCounter);
            }
    }
```

Taint Block

TAINT CAPTURED

Use static analysis to identify all taint blocks and instrument the app binary automatically.

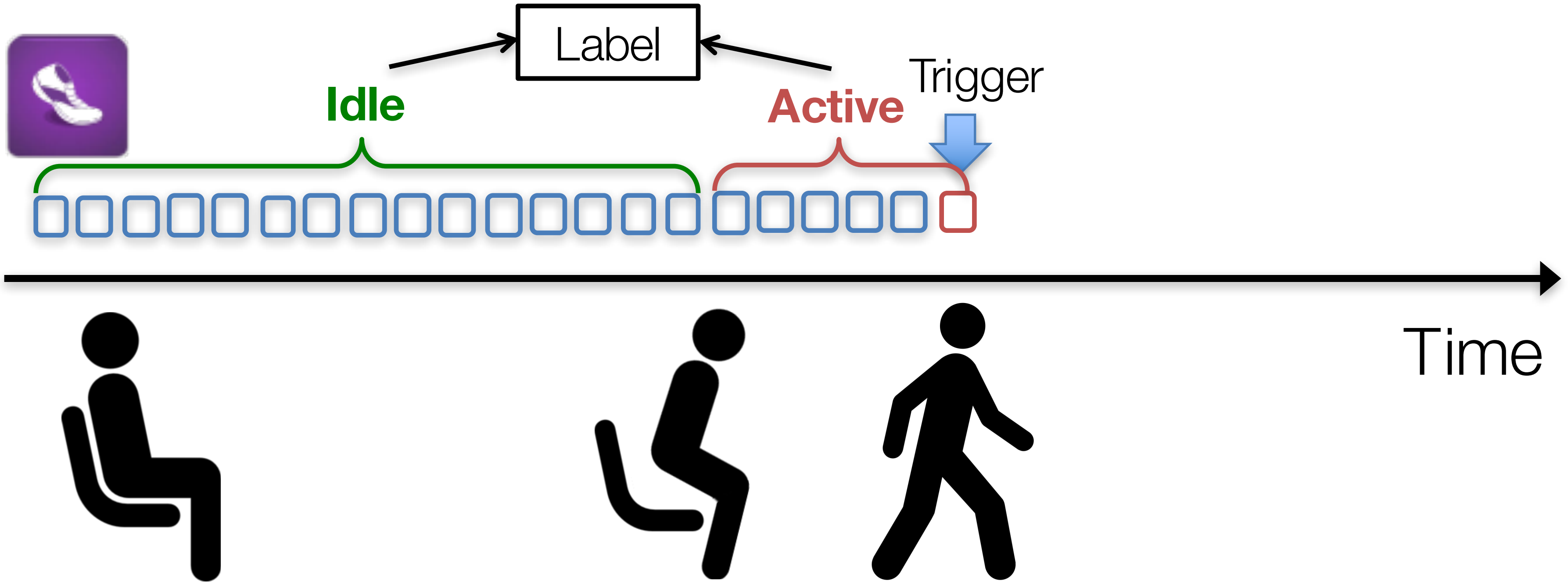
MobileHub system overview



Step 0: Sensor traces
Original app
Optimized app
Step 1 Dynamic taint tracking: Track app notifications for a series of sensor inputs
Taint log
Step 2 Learning: Learn how changes in sensor values result in app notifications
Sensor parameter
Classifier model
Step 3b Application rewriting: Rewrite app to offload sensing to the sensor hub
Step 3a Sensor hub program: Implement the classifier in the sensor hub, corresponding to the app
MobileHub System

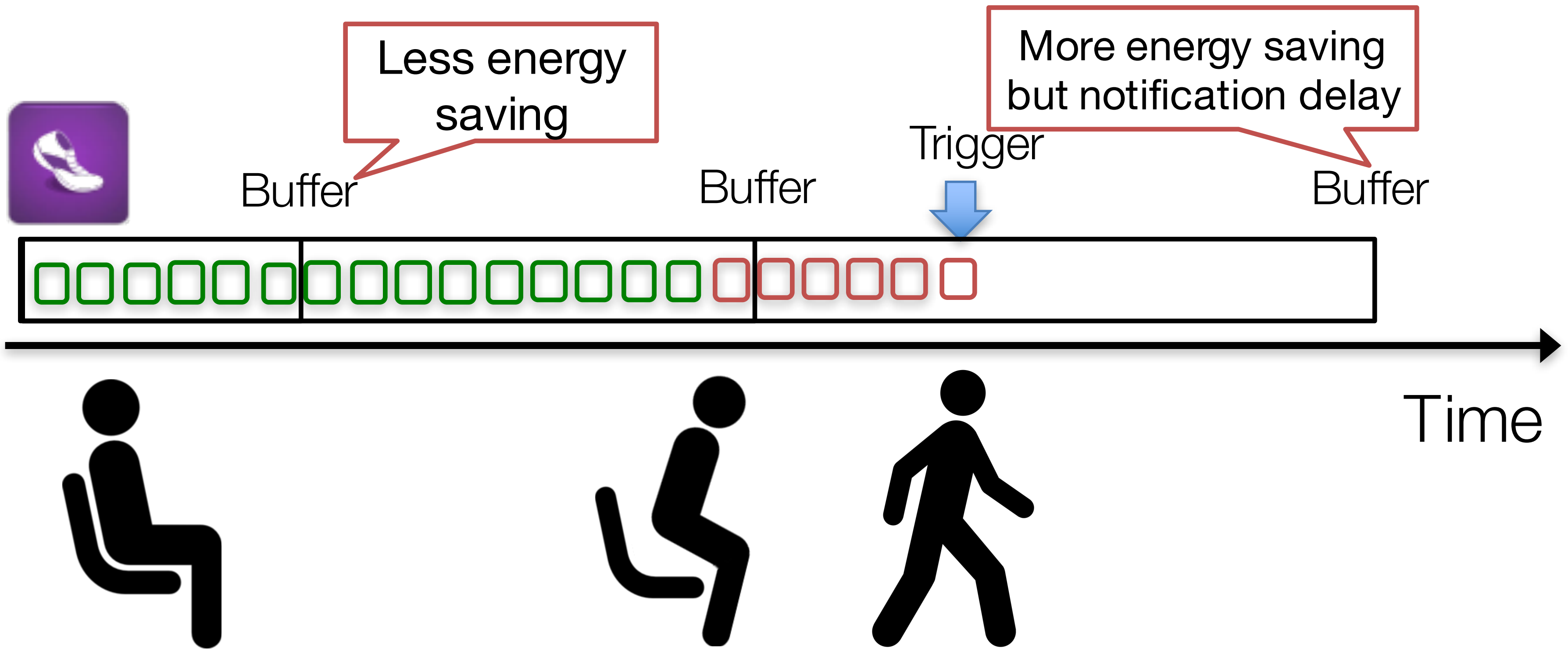
Learning a buffer policy

- Hard to use a classifier to model the app logic
- Simply learn the statistical properties and distinguish between idle and active periods

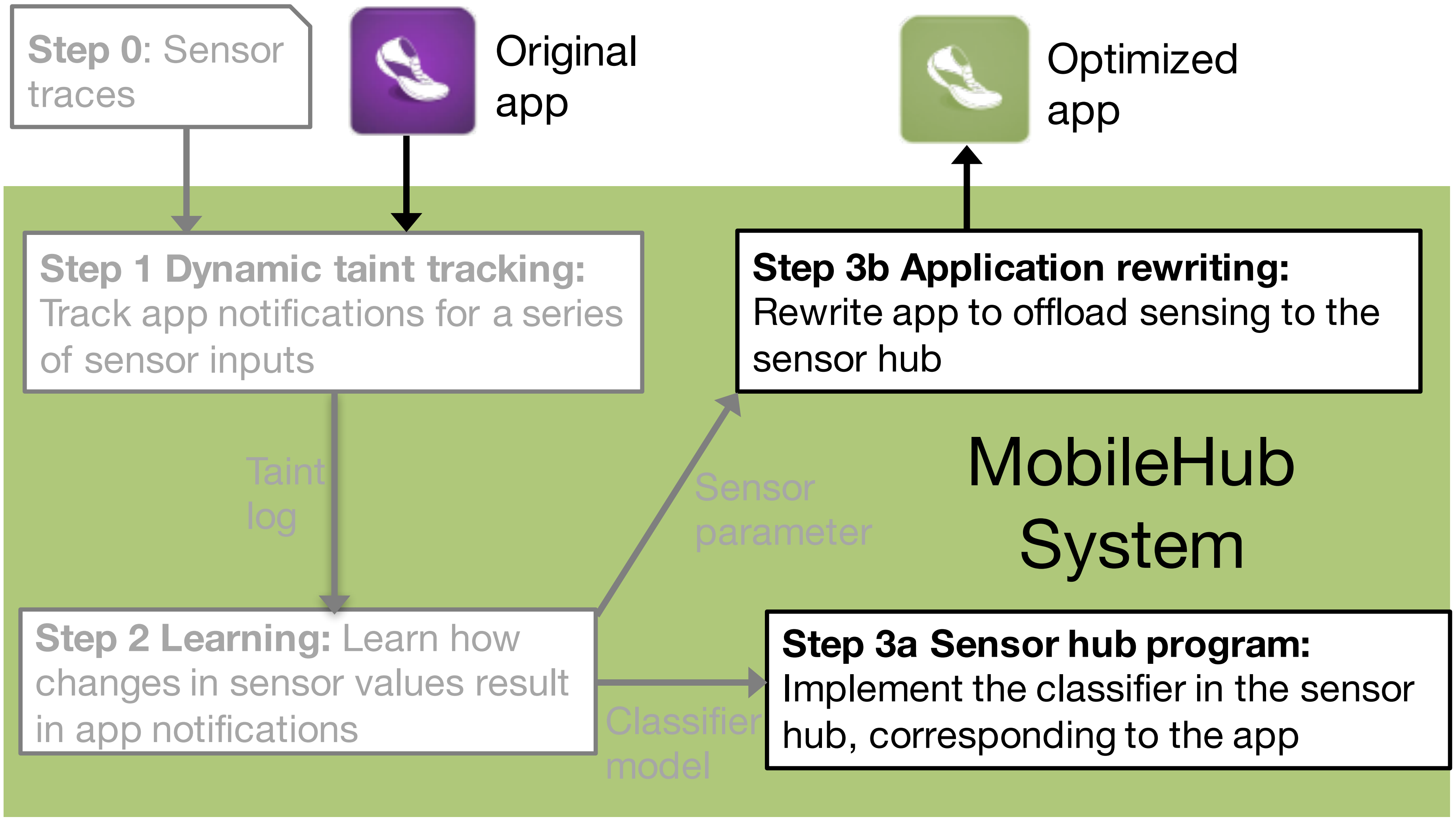


Goal: find a proper buffer size

- Predict active and idle periods
- Reduce the number of notification delays



MobileHub system overview



Step 0: Sensor traces

Step 1 Dynamic taint tracking: Track app notifications for a series of sensor inputs

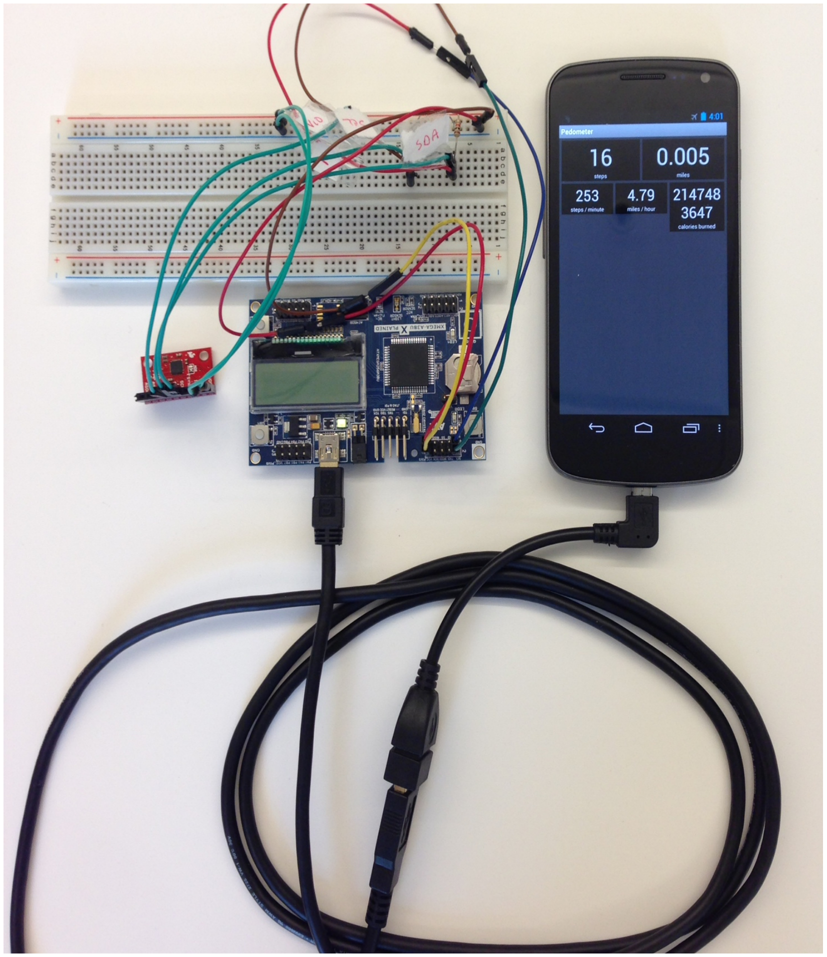
Step 2 Learning: Learn how changes in sensor values result in app notifications

Step 3a Sensor hub program: Implement the classifier in the sensor hub, corresponding to the app

Step 3b Application rewriting: Rewrite app to offload sensing to the sensor hub

Implementation

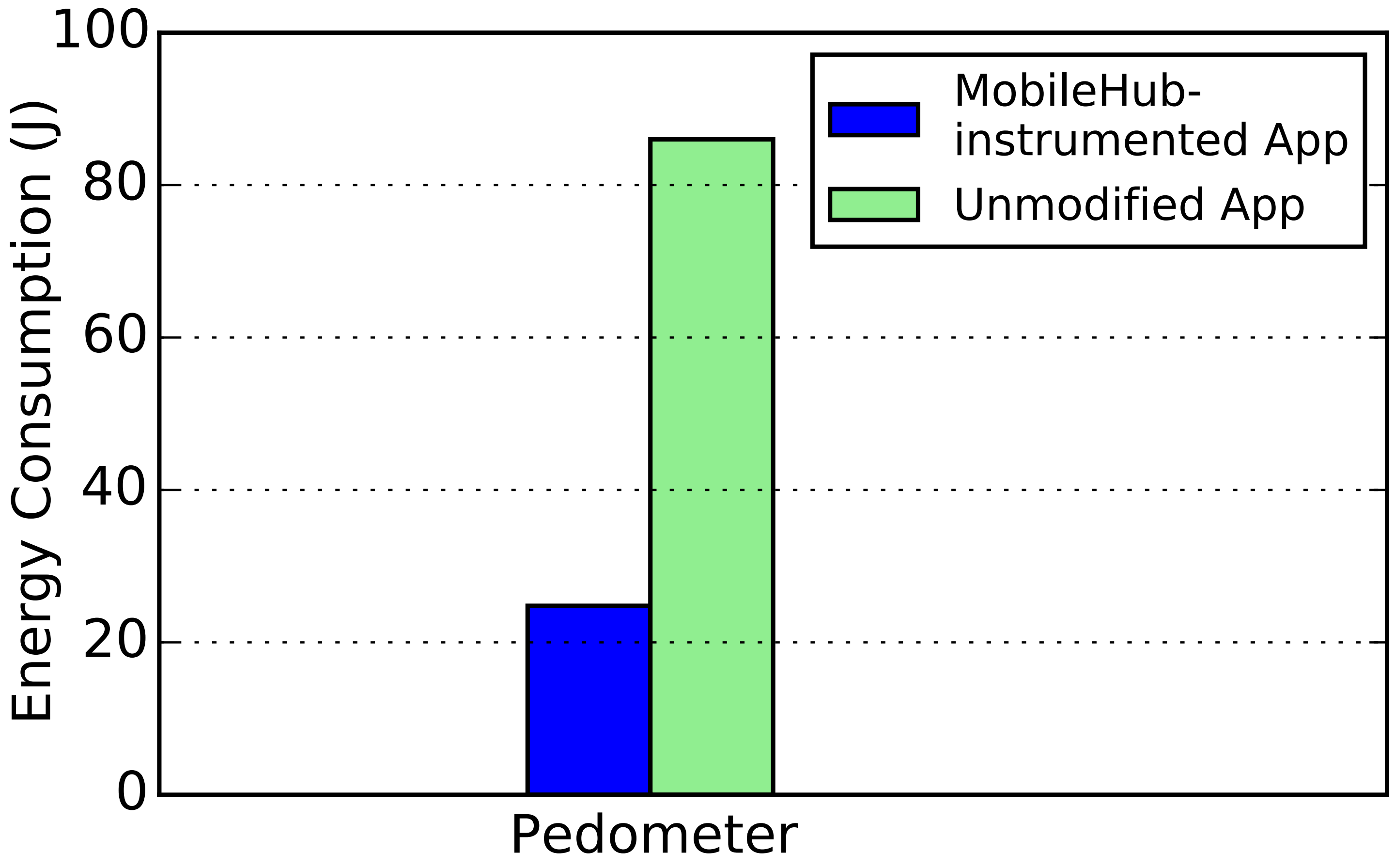
- Implemented in Android
  - Taint tracking system
  - Interface with sensor hub
  - App binary rewriter
- Prototype
  - Implemented classifier on sensor hub



Evaluation

- Does the prototype work?
- Does MobileHub improve power consumption on real traces?
- Does MobileHub work for a large number of apps?

Prototype measurement



Evaluation using real sensor traces

- Trace collection from 21 participants
  - 10 traces for sleeping, driving, and daily life
  - 5 traces for other activities

- Downloaded 20 apps from Google Play

Name	Google Play Store ID	Task	Sensor
nWalk	pl.rork.nWalk	Step counting	Accelerometer
pedometer	bagi.levente.pedometer	Step counting	Accelerometer
stepcounter	Stepcounter.Step	Step counting	Accelerometer
appsone	net.appsone.android.pedometer	Step counting	Accelerometer
virtic	jp.virtic.apps.WidgetManpok	Step counting	Accelerometer
walking	cha.health.walking	Step counting	Accelerometer
lodecode	com.lodecode.metaldetector	Metal detector	Magnetometer
imkurt	com.imkurt.metaldetector	Metal detector	Magnetometer
tdt	com.tdt.magneticfielddetector	Metal detector	Magnetometer
multunus	com.multunus.falldetector	Fall detector	Accelerometer
iter	com.iter.falldetector	Fall detector	Accelerometer
t3lab	it.t3lab.fallDetector	Fall detector	Accelerometer
fall	com.fall	Fall detector	Accelerometer
jietusoft	com.jietusoft.earthquake	Earthquake detector	Accelerometer
vibration	ycl.vibrationsensor	Earthquake detector	Orientation
posvic	cz.posvic.fitnessbar.sleeptrack	Sleep monitoring	Gyroscope
myway	myway.project.sleepmanagement	Sleep monitoring	Accelerometer
driving	jp.co.noito.Accelerometer	Driver monitoring	Accelerometer
motion	com.app.accelerometer	Motion detector	Accelerometer
thefthead	com.thefthead.appfinalsettings	Theft detector	Accelerometer

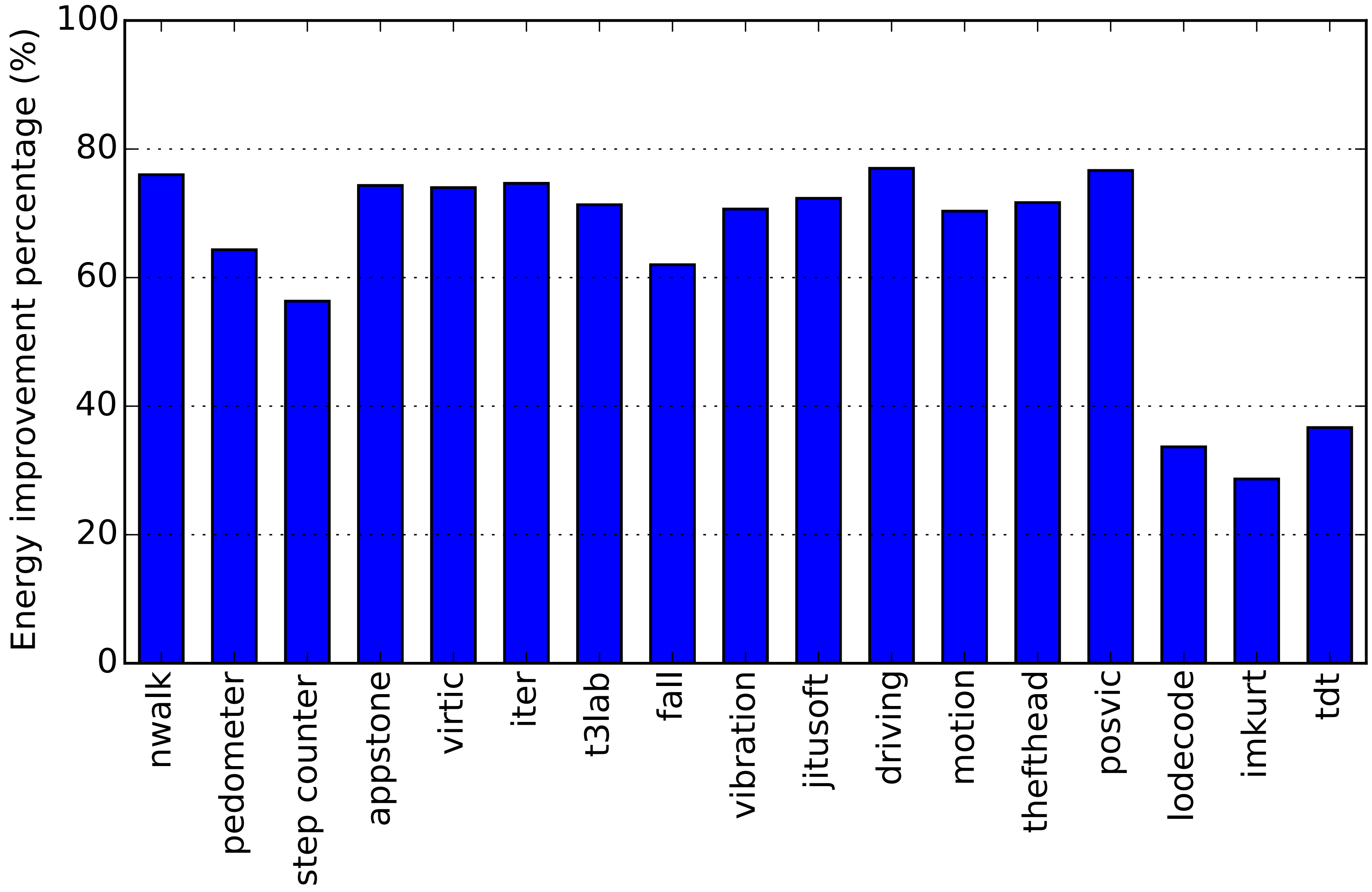
Trace evaluation methodology

- Run each app on the phone receiving sensor values from a trace file
- Trace file embeds the buffering policy

Power Accounting:

- Measure the power consumption of phone
- Deduct the standby power consumption

Energy improvement



Notification delay

- Notification is delayed by at least 0.5s

App	Task	#Delay/#Notifications	Max delay (s)
nWalk	Step Counting	1/3914	1.86
imkurt	Fall Detection	2/142	0.98
posvic	Sleep Monitor	1/36	0.64
thefthead	Anti-theft	6/65	2.80

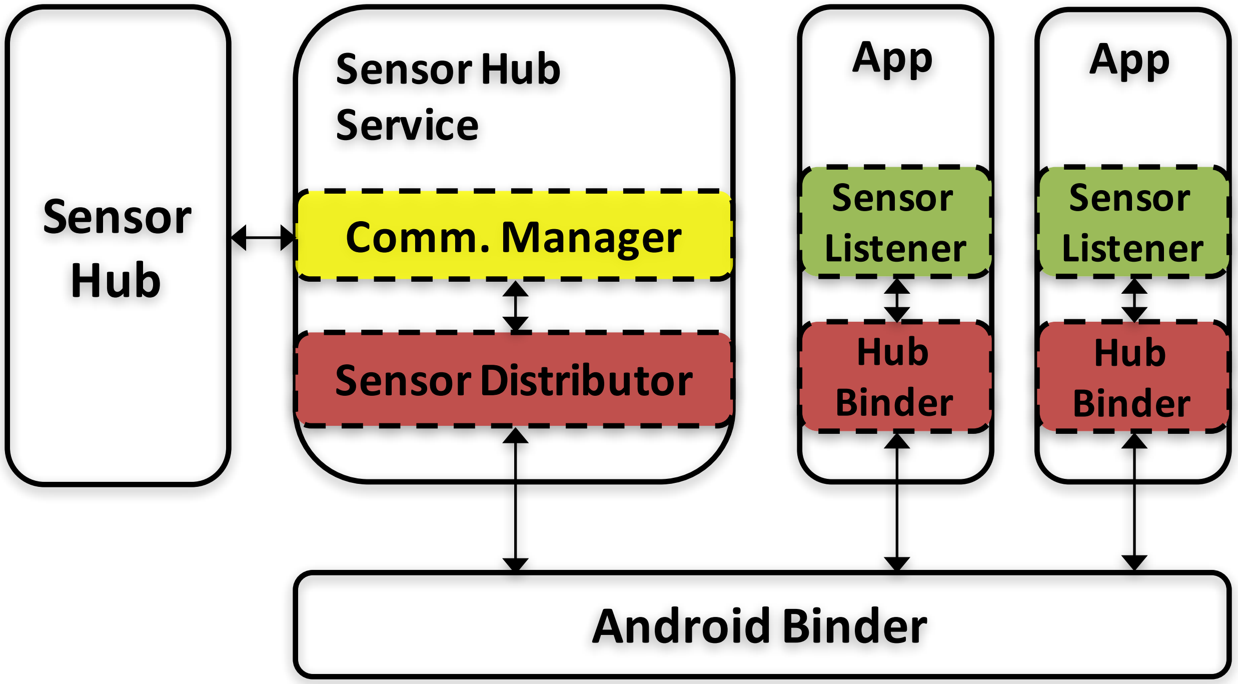
Conclusion

- Design and implement MobileHub that rewrites application to leverage sensor hub without programmer effort
- Experiment with 20 sensing apps, and reduce power consumption by 74% in median
- MobileHub delays 1.5% app notifications across all apps on average

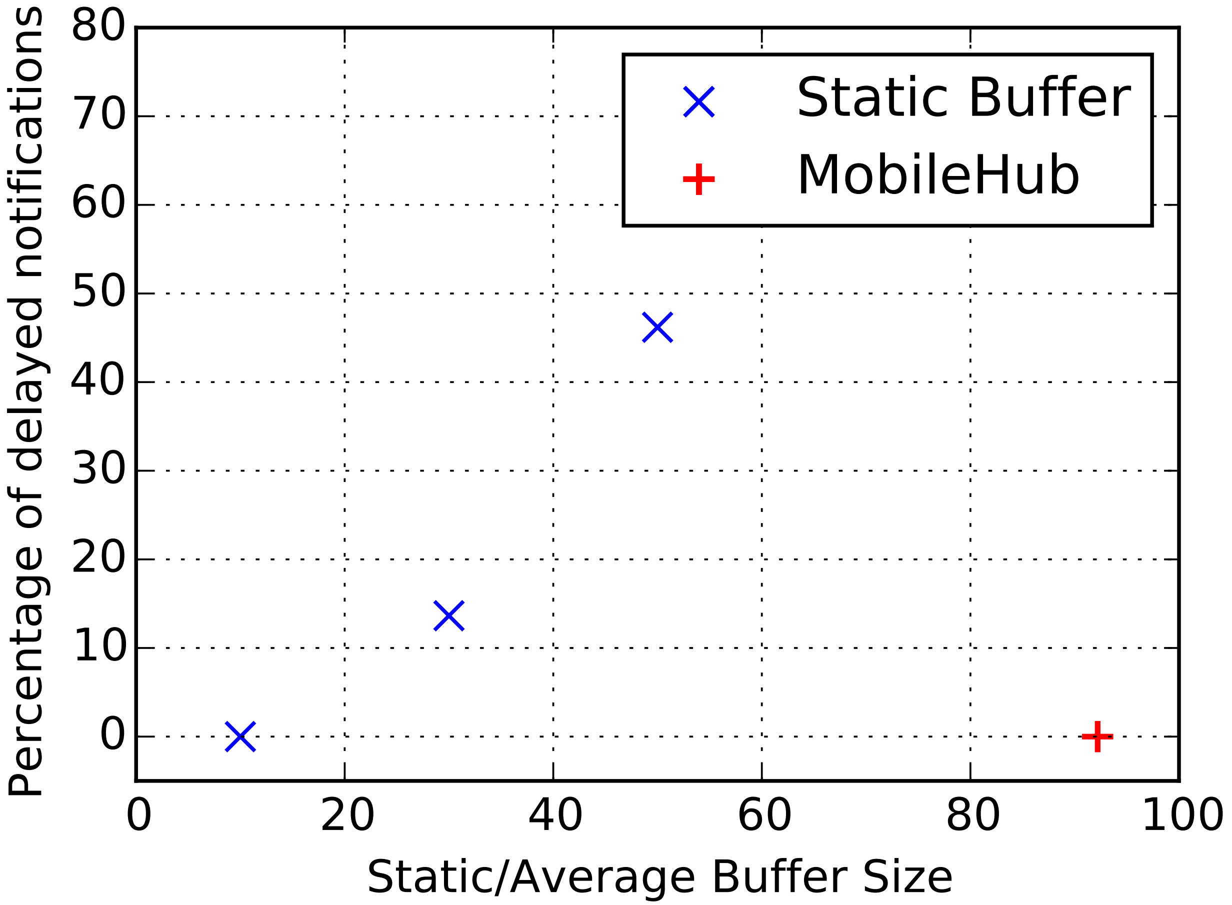
Thank you!

haichen@cs.washington.edu

Sensor Hub Service



Dynamic vs static buffer


Percentage of delayed notifications
Static/Average Buffer Size
Static Buffer
MobileHub